NAVAL POSTGRADUATE SCHOOL MONTEREY CALIF
A REAL-TIME OPERATING SYSTEM FOR SINGLE BOARD COMPUTER BASED DI--ETC(U) AD-A059 601 JUN 78 W NIEMANN UNCLASSIFIED NL 1 of 4 AD A059 601

AD AO 59601



# NAVAL POSTGRADUATE SCHOOL Monterey, California

DOC FILE COPY.



DDC oct 11 1978

# THESIS

A REAL-TIME OPERATING SYSTEM FOR SINGLE BOARD COMPUTER PASED DISTRIBUTED NAVAL TACTICAL DATA SYSTEMS

Wolfgang/Niemann

June 1978

June 1978

Professor U. R. Kodres

Approved for public release; distribution unlimited.

251 450

JUB

78 10 06 048

SECURITY CLASSIFICATION OF THIS PAGE (When Date Entered)

REPORT DOCUMENTATIO	READ INSTRUCTIONS BEFORE COMPLETING FORM				
NEPONY NUMBER	S. SOUT ACCESSION NO.	3. RECIPIENT'S CATALOG HUMBER			
A Real-time Operating System for Computer Based Distributed Nava	8. TYPE OF REPORT & PERIOD COVERS Master's Thesis; June 197				
Systems.	6. PERFORMING ORG. REPORT NUMBER				
T. AUTHOR(4)	S. CONTRACT OR GRANT HUMBER(s)				
LT Wolfgang Niemann					
S. PERPORMING ORGANIZATION NAME AND ADDRE	10. PROGRAM ELEMENT, PROJECT, TASK				
Naval Postgraduate School Monterey, California 93940					
Naval Postgraduate School	June 1978				
Monterey, California 93940	13. NUMBER OF PAGES 346				
TE. MONITORING AGENCY HAME & ADDRESS(IF MITTE	rent from Controlling Office)	18. SECURITY CLASS. (of this riport)			
Naval Postgraduate School	Unclassified				
Monterey, California 93940	The DECLASSIFICATION/DOWNGRADING				
A SIEVENINIAN SVAVENENS (ALAMA BARAN)					

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the obstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Distributed computer systems, Single Board Computers, Naval Tactical Data Systems, real-time operating system

20. ABSTRACT (Continue on reverse side it necessary and identify by block number)

The microprocessor revolution has produced a capable computer on a single printed circuit board.

The design and development of a real-time operating system for a distributed system of Single Board Computers is presented in this paper.

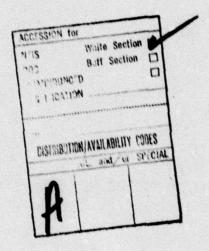
There are user manuals and program descriptions for the operating system, a debug module, a CRT module and a line printer module.

The operating systems has been developed for a Multibus system with three INTEL Single Board Computers SBC80/20-4 and 64K bytes of common memory

DD 1 JAN 73 1473 EDITION OF 1 HOV 68 IS OBSOLETE

DECUMTY CLASSIFICATION OF THIS PAGETYMEN POIL BRICKS

The system has been designed specifically for Naval Tactical Data Systems applications and the feasibility of such applications are evaluated with respect to currently available Single Board Computers and with respect to Single Board Computers that should be available in the near future.



Approved for public release; distribution unlimited.

A REAL-TIME OPERATING SYSTEM
FOR
SINGLE BUARD COMPUTER BASED
DISTRIBUTED
NAVAL TACTICAL DATA SYSTEMS

by

Wolfgang Niemann Lieutenant, Federal German Navy

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 1978

Author:	Wolfe	an h	ren er	<u> </u>
Approved by:	Uno	P. E	odre	)
	Be	uphear	T	hesis Advisor
	Ma		M	Second Reader
	Chairman	1 Sth	of com	puter Science
	Dean of I	nformation	Po	licy Sciences

#### ABSTRACT

The microprocessor revolution has produced a capable computer on a single printed circuit board.

The design and development of a real-time operating system for a distributed system of Single Board Computers is presented in this paper.

There are user manuals and program descriptions for the operating system, a debug module, a CRT module and a line printer module.

The operating system has been developed for a Multibus system with three INTEL Single Board Computers SBC80/20-4 and 64K bytes of common memory.

The system has been designed specifically for Naval Tactical Data Systems applications and the feasibility of such applications are evaluated with respect to currently available Single Board Computers and with respect to Single Board Computers that should be available in the near future.

# TABLE OF CONTENTS

1	INTRODUCTION	8
11	COMPUTING REQUIREMENTS FOR NAVAL TACTICAL DATA	
	SYSTEMS	10
111	A COMPUTER SYSTEM USING SINGLE BOARD COMPUTERS	13
	A. INTEL'S SINGLE BOARD COMPUTER SBC80/20-4	13
	B. SYSTEM CONCEPT	16
	C. SYSTEM BUS ORGANIZATION	18
	D. MEMORY ORGANIZATION	19
	E. INPUT/OUTPUT FACILITIES	21
	F. INTERRUPT STRUCTURE	25
14	A REAL-TIME OPERATING SYSTEM FOR SINGLE BOARD	
	COMPUTERS	24
	A. SYSTEM CONCEPT	25
	B. MODULAR STRUCTURE	56
	C. FACILITIES OF THE OPERATING SYSTEM	26
	1. Priority Tasks	56
	2. Communication between Modules	27
	3. Time Dependent and Periodic Tasks	28
	4. Background Tasks	29
	5. System Calls	59
	6. Real-time Clock and Count Down Clock	29
	D. REAL-TIME EXECUTIVE	30
	F. INTERRUPT HANDLING	31

	F.	SYSTE	EM M	TINO	OR.	• • • •	• • • •	• • • •	• • • •	••••	••••	• • • •	••	35
	G.	SYST	EM I	NTEG	RAT	ION.	• • • •		••••	••••	•••		••	35
٧	AVA	ILABI	LE U	SER	MODI	JLES	• • • •			••••	•••		••	34
	۸,	CRT I	HODU	LE	•••	• • • •	• • • •			••••	•••		••	34
	в.	LINE	PRI	NTER	MOL	PULE	• • • •				• • • •		••	35
	c.	DEBU	5 MO	DULE	•••				• • • •				••	36
٧I	CON	CLUS	IONS	••••	•••		• • • •				•••		••	36
	٨.	HARDI	NARE	DES	IGN.		• • • •			••••	•••		••	37
		1.	Stan	dard	izat	tion	and	Cost		••••	•••			37
		2. (	Expa	ndab	1111	y	• • • •		••••		•••		••	37
		3.	Inte	rfac	e w	ith I	Perio	hera	1 Ec	uipm	ent		٠.	38
		4.	Main	tena	nce	and	Rel	abil	ity.	••••	• • • •		••	38
		5.	Phys	ical	Red	auir	emeni	·	••••	••••	•••		••	38
		6.	Syst	em R	edur	ndani	cy		••••		•••		••	39
	в.	OPER	ATIN	G SY	STE	٧	• • • •			•••	•••		••	39
		1.	Nava	1 Ta	cti	cal	Data	Syst	em F	e au i	rem	ents	•••	39
		2.	Soft	ware	De	velo	pmen	. Mai	nter	ance	and	d		
			Exte	nsio	ns.	••••	• • • •			••••			••	40
		3.	Stan	dard	izal	tion	• • • •			••••			••	40
		4.	Simu	lati	on.	• • • •	• • • •			••••	• • • •		••	41
API	ENDI	CES:												
	A -	Prog	ram	Desc	ripi	tion	for	Oper	atin	g Sy	ster	n <b></b>	••	42
	в -	User	's M	lanua	1 60	or 01	peral	ing	Syst	em.			••	67
	c -	User	's M	lanua	1 +	or D	ebug	Func	tion	·s			••	95
	0 -	Prog	ram	Desc	ripi	tion	for	Dehu	g Mo	dule			1	08
	E -	Prog	ram	Desc	ripi	tion	for	CRT	Modu	1e			1	30
	F -	Peac		Dasc	cin		100	Line	Pri		Mod	1111	1	49

G	•	Program	List	ings	 	 	 	 .160
-		TO THE PERSON NAMED IN COLUMN TWO IS NOT THE OWNER.						

# I. INTRODUCTION

An examination of currently installed Naval Tactical Data Systems reveals that the heart of the system, the computer, does not represent today's advanced technology. Even in recently implemented systems large 'space, weight, power and maintenance cost consuming' second generation computers are found. The use of second generation technology is caused by lengthy lead times in systems acquisition, system conversion and especially software conversion cost, educational cost etc. However, in the age of microcomputers, which provides features like low hardware cost, high degree of versatility and therefore a potential for standardization, high reliability, low maintenance cost and low power, space and weight consumption, it should be the time to develop new systems in order to make use of these features which seem to be tailored specifically for military applications.

The real-time operating system developed in this thesis should be understood as a step in the direction of making use of the new technology. The operating system is designed for a distributed system of concurrently operating Single Board Computers.

After identifying typical computing requirements for Naval Tactical Data Systems, a distributed computer system using Single Board Computers and a real-time operating system are developed. Three user modules, a debug, CRT, and a line printer module, are introduced. In the conclusion of this paper a comparison between the identified requirements and the developed system is made and possible extensions are indicated.

# II. COMPUTING REQUIREMENTS FOR NAVAL TACTICAL DATA SYSTEMS

In this section the basic requirements for a computer system which is to drive a Naval Tactical Data System are considered.

In general it can be said that the computer system has to be able to handle the workload dictated by the operational specifications and to cope with peak situations without a system failure.

Typically, Naval Tactical Data Systems are dedicated systems. Because of this fact, system requirements for the CPU, memory and input/output are known. It is therefore not necessary to carry a vast amount of overhead in order to be prepared for unknown worst cases. However, when deciding on a computer system, future extensions of the system with hardware consequences should be taken into account. Unfortunately it is very difficult to predict the possible enhancements during the life time of a Naval Tactical Data System. Therefore, the chosen computer system should be expandable.

Since Naval Tactical Data Systems are rather complex systems with many different system functions, the equipment used in an implementation is produced by many different manufacturers. Although there are some standard interfaces

for Naval Tactical Data System, the computer, which is to drive the peripheral hardware has to be versatile in order to be connected to different devices with different interfaces.

In order to reduce cost by large series and simplified maintenance a standardization between different systems is highly desirable.

The instruction repertoire of the computer system has to provide instructions which allow the efficient programming of typical operations in Naval Tactical Data Systems. Typical operations are

- the solution of complex mathematical problems
   in a reasonable time (quasi real-time)
- bit manipulations
- fast data base access
- complex and fast input/output operations
- extensive interrupt handling.

Many command and control operations and decisions depend on the proper functioning of the Naval Tactical Data System. High reliability, even under extreme physical conditions, are therefore a top requirement for this kind of system. In case of a breakdown, the tactical information often is lost or it takes some time to recreate a valid tactical situation. Because of the importance of the Naval Tactical Data System within a larger system (ship or

aircraft), a complete back-up for the computer system is desirable.

The major requirement to be met by the operating system is to run the system under real-time conditions. Dependant on the computer system, this leads to operating systems of differing sizes where the ratio of time used by the operating system to total time should be optimized.

Basically, the operating system has to support an overall program structure which simplifies

- software development
- software maintenance
- software extensions
- use of components from other systems
   (standardization).

#### III. A COMPUTER SYSTEM USING SINGLE BOARD COMPUTERS

In this section the hardware concept of a computer system consisting of Single Board Computers is developed. Basic building block of this computer system is a INTEL Single Board Computer, SBC80/20-4.

# A. INTEL'S SINGLE BOARD COMPUTER SHC80/20-4

INTEL's Single Board Computer, SBC80/20-4, represents a complete microcomputer on a single printed circuit board.

#### The SBC80/20-4 includes:

- 8080A CPU
- 4K static Random Access Memory (RAM)
- up to 8k Read Only Memory (ROM)
- 48 programmable parallel input/output lines
- a programmable serial input/output interface
- programmable interval timers
- programmable, eight priority level, vectored interrupt structure
- bus interface for external system bus.

An in-depth description of hardware, function and programming of the SBC80/20-4 is given in the SBC80 manual [INTEL SBC80/20 HARDWARE REFERENCE MANUAL 98-317c].

# 1. 8080A CPU

The 8080A is a single LSI chip CPU. It has six 8-bit general purpose registers and an accumulator. The six general purpose registers may be addressed individually or as register pairs, providing 16-bit operations.

A 16-bit stack pointer controls the addressing of an external stack which can be located in general memory.

The 8080A has an address range of 64K bytes.

The CPU Set consists of the 8080A CPU, clock generator and a system controller. It performs all system processing functions and provides a stable timing reference for all other circuitry on the board.

# 2. Random Access Memory/Read Only Memory

The 4 K Random Access Memory on the Single Board Computer can be jumper assigned to the top address space in any one of the four 16K address blocks.

The Read Only Memory is located starting at address 0000 and has a size of 4K or 8K depending on the type of memory devices used.

The full CPU capability of addressing 64k bytes can be utilized by adding external memory boards which are accessible via the system bus. The respective parts in this memory are 'shadowed' by the on-board memory. The CPU set is capable of determining whether an addressed

memory location is on-board or not.

#### 3. Parallel Input/Output

The SBC provides 48 input/output lines which can be configured by software in combinations of uni-directional or bi-directional input/output ports.

# 4. Serial Input/Output

The SBC includes a programmable synchronous/asynchronous RS232C communication interface which is capable of operating with all common communication frequencies.

# 5. Interval limers

The SBC contains two fully programmable and independent BCD or 16-bit binary interval timers/event counters.

# 6. Interrupt Structure

The on-board interrupt controller provides vectoring for up to eight interrupt leves in four different priority processing modes. Operating mode and priority assignment are under software control.

# 7. System Bus Interface

This interface is compatible with INTEL's Multibus system and allows the combination of several Single Board

Computers, memory and other utility boards on one system bus.

Two bus priority systems are available: serial (up to three master controller) and parallel (up to sixteen master controller).

#### B. SYSTEM CONCEPT

The development of a computer system consisting itself of rather independent Single Board Computers automatically leads to the concept of a distributed system. A distributed system in this context is a computer system in which several processors are working on more or lass independent tasks connected by a common system bus. The computer system to be developed is a direct realization of this concept.

Basically the system consists of Single Board Computers and 64K bytes of Random Access Memory external to the Single Board Computers. The external memory resides on four printed circuit boards which are bus compatible with the Single Board Computers. This memory represents a common memory to all Single Board Computers attached to the same bus. This concept in connection with the on-board memory of the Single Board Computers opens up software possibilities which are explored in Chapter IV.

Information interchange between Single Board Computers can be realized using three different concepts.

#### Concept (1):

Storage of information in common memory and periodic checks of the agreed upon 'mail boxes'. This concept can be used in systems where all processing is organized in a hierarchical 'producer - consumer' structure. In this structure basic processing is local to a Single Board Computer. Data is gathered and processed at a high rate. The output, reduced data and updates of common data bases, is required by others processors at a lower frequency. Since external events can occur asyncronously, an extra data path through common memory for such messages has to be provided.

#### Concept (2):

Storage of information in common memory and interruption of the information receiving Single Board Computer. The addition of interrupts to concept (1) allows both synchronous and asynchronous transfers of data sets between Single Board Computers. The disadvantage of this concept is the number of interrupt lines required with an increasing number of participating Single Board Computers in order to address each other. The alternative of having only one interrupt common to all Single Board Computers would cause an interruption of all processors and requires a polling scheme to determine the receiving Single

Board Computer.

#### Concept (3):

Direct transfer of information between Single Board Computers using input/output ports and interrupts. This concept can be used to exchange time critical information between processors. Since it directly involves the CPU of both the sender and receiver the amount of data passed has to be kept small. However, large data sets can be transferred with the use of pointers to common memory. The limitations of concept (2), extended for data lines between input/output ports, also apply for concept (3).

The input/output structure of the Single Board Computers allow the connection of a variety of peripheral devices to the computer system. Each connection represents a hardware modification or specialization of a Single Board Computer, i.e. the dedication of a part of the computer system to a special task.

#### C. SYSTEM BUS ORGANIZATION

The system bus, which is the main communication line between several Single Board Computers, common memory and other utility printed circuit boards, is implemented using INTEL's Multibus. This bus system allows master-master and master-slave relationships between various system hardware modules.

Transfers via this bus proceed asynchronously, i.e. the transfer speed is dependent on the speed of the transmitting and receiving devices. Once a module has gained control of the bus, transfers can proceed with a maximum rate of 5 million bytes/second.

Master modules can gain access to the bus using a serial or parallel priority resolving scheme. The serial scheme is limited to three masters because of the signal propagation delay, but up to sixteen masters may be connected to the bus using the parallel priority scheme.

The bus system provides an override facility which allows a hardware module to keep control of the bus until the operation is completed. This feature can be used under software control.

#### D. MEMORY ORGANIZATION

The total memory of the computer system consists of

- common Random Access Memory
- on-board Random Access Memory
- on-board Read Only Memory.

The on-board Random Access Memory portions are located in the same region on all Single Board Computers. This leaves the respective portion in common memory unused, however, it is now possible to run identical programs on the Single Board Computers. Identical programs allow the access

of common data and execution of shared code. The location of on-board Random Access Memory can be changed because all programs are relocatable.

All variable data has to be kept in on-board memory for two reasons:

Faster access by CPU and increased overall
performance because use of the system bus is
avoided.
 On-board memory access does not require WAIT states

of the CPU.

2) No data conflicts in the execution of shared code. Although the variable data has the same logical address space, its physical representation is local to the Single Board Computers and therefore 'invisible' to other CPUs.

On-board Read Only Memory contains the executive and frequently executed program parts. Other program parts, especially when they are identical in all Single Board Computers are located in common memory in order to allow shared execution.

#### E. INPUT/OUTPUT FACILITIES

Each Single Board Computer provides serial and parallel input/output facilities which can be completely driven by interrupts.

The serial interface is ready to be used with a CRT. with the addition of an adapter a TTY can also be connected to a Single Board Computer.

Each Single Board Computer provides six bi-directional 8-bit ports. These ports can be configured by software to be 8-bit data input/output ports or 4-bit bit addressable input/output ports. The latter can be used to receive and send status bit information in conjunction with the 8-bit input/output ports.

The hardware provides three basic modes of operation that can be selected by software:

- Mode 0 : Basic Input/Output

- Mode 1 : Strobed Input/Output

- Mode 2 : Bi-directional Bus

Mode 0 can be used for simple, status driven device interfaces. Since this kind of interface is not compatible with the real-time requirements it is not considered here.

Mode 1 and Mode 2 generally require the support of interrupts. Mode 1 provides a means for transferring data to or from a specific port in conjunction with strobe or

'hand-shake' signals. This mode can be used for a variety of different peripheral devices.

Mode 2 provides communication with a peripheral device on a 8-bit bus for both receiving and transmitting data. 'Hand-shake' are provided to maintain proper bus flow in a similar manner to Mode 1.

The driver/termination connections of the parallel input/output section are left open and have to be inserted depending on the actual implementation.

The primary user considerations in determining how to use each of the six input/output ports are:

- choice of operating mode
- direction of data flow
- choice of driver/terminator networks
- jumper configurations
- mutual port restrictions.

# F. INTERRUPT STRUCTURE

The interrupt controller accepts jumper selectable interrupt requests from

- parallel input/output
- serial input/output
- interval timers
- system bus

# - directly from external devices.

The controller resolves priority among the eight possible interrupt levels according to an algorithm which is selectable by software. The priority assignments and algorithms can be changed dynamically at any time during system operation.

Of the four possible interrupt modes only the 'fully nested' mode is considered here. In this mode priorities are fixed such that level 0 has the highest priority and level 7 the lowest.

The interrupt hardware provides vectored interrupts where the interrupt vector is not fixed in its location and size (4 or 8 bytes/interrupt). The interrupt controller has to be programmed with location and step width of the interrupt vector.

The interrupt controller allows the setting of a one byte interrupt mask by software. This feature allows the inhibition of not wanted interrupts in any combination of the eight interrupt levels. IV. A REAL-TIME OPERATING SYSTEM FOR SINGLE BOARD COMPUTERS

The objectives for the operating system to be developed in this section are:

- 1) The operating system has to be able to control NIDS applications under real-time conditions.
- 2) The host computer system is the multi processor microcomputer system described in the previous section.
- 3) The existing Microcomputer Development System and the associated support software (ISIS-II,PL/M-80) has to be used for program development.
- 4) The operating system must provide debugging tools that allow system debugging and system testing under real-time conditions.

This chapter describes the operating system in more general terms. An in-depth description is given in the user's manual (Appendix A) and in the program description of the operating system (Appendix B).

A TASK is a part of the program which handles a specific function of the system and consists of one or more procedures.

A MODULE is a separate task or separate group of related tasks which are considered to be independent in terms of software development and system integration.

The EXECUTIVE is the kernel of the operating system and controls the scheduling and execution of tasks.

The OPERATING SYSTEM consists of executive, system calls and system data.

#### A. SYSTEM CONCEPT

In contrast to an operating system for general usage in which the nature of the running tasks is not predictable this real-time operating system drives a dedicated system. Dedicated, in this context, means that the implemented tasks do not change at run time of the system. This concept allows dropping much of the book-keeping overhead which is typical for operating systems for general usage. Tasks in this system are not physically moved in memory, they are only activated and suspended.

Execution of tasks is under control of the kernel of the operating system, the executive. The executive controls the CPU assignment to the various tasks activated by interrupts, to process a message from another task or at a preset point in time.

#### B. MODULAR STRUCTURE

The operating system supports a strictly modular program structure. Modules are integral parts of the overall program which usually handle a specific function or a group of related functions. This organization not only eases program development and maintenace, it also allows easy-to-implement changes of the system. Typically, a module is compiled separately and then integrated into the rest of the system. Because of the independent nature of modules they can be removed or replaced without influencing the remaining system. Not only user programs represent modules, the operating system itself consists of independent modules.

Modules are identified by a number. The number depends on the number of the Single Board Computer which hosts the module. Maximum number of modules per Single Board Computer is 8. Modules in Single Board Computer 1 have numbers 0 to 7, in Single Board Computer 2 numbers 8 to 15 etc. The lowest module number in a Single Board Computer is reserved for the executive of the operating system while the highest number represents the debug module.

#### C. FACILITIES OF THE OPERATING SYSTEM

#### 1. Priority Tasks

Priority tasks represent the highest priority level a task can have in the system. Priority tasks are executed as soon as processor control is returned from the current

active task.

A priority task has to be entered into the list of priority tasks with a system call. A single execution of that priority task can then be scheduled with another system call.

The primary purpose of priority calls is the process of interrupts. The call of the priority task has to be scheduled in the interrupt service routine. The high priority of that task ensures that it is executed as soon as the processor becomes available.

# 2. Communication between Modules

Since modules are separate and independent parts of the system the operating system has to provide a function which allows the tasks or modules to communicate with each other. This communication uses the form of messages which are sent from one module to another. A message is sent with a system call and entered into a FIFO list. The receiving module's message entry is called if no priority task is pending and the message is to be processed next in the FIFO list.

A message consists of message control block and possibly data bytes. The message control block identifies receiving module, sending module, message number and length of the message. The message number depends entirely on an agreement between sending and receiving module and serves

the purpose of identifying the message. If the message control block itself is not sufficient for the transmission of information, data bytes can be added to the message.

A message is always sent from one module to another module regardless in which Single Board Computer they reside. The operating system decodes the number of the receiving module and routes the message to another Single Board Computer if necessary.

#### 3. Time Dependent and Periodic Tasks

Real-time environments and especially Naval Tactical Data Systems require the execution of certain tasks at a predefined point in time or periodically with a specified time interval. In this operating system these tasks range in the priority hierarchy below the priority tasks and the process of messages.

Periodic tasks have to be identified to the operating system with a system call. With this system call the task and the specified time interval is entered into the list of periodic tasks. The executive uses the system's real-time clock to determine the exact time of the call. A periodic task can be suspended or the specified time interval can be changed with system calls.

# 4. Background Tasks

Background tasks represent the lowest priority level in the system. They are executed only if no priority task is pending, no message is to be processed and no periodic call is necessary, i.e. the processor is idling. This idle time can be used to perform hardware checks on a time slice basis or to perform data reductions for statistic and test purposes.

If a Single Roard Computer is completely interrupt driven, (i.e. not periodically activated) the processor can enter the HALT state to free the system bus. The next interrupt will 'awake' the processor and the executive.

#### 5. System Calls

The operating system provides system calls for task management, communication between modules and functions which are commonly used by several modules.

#### 6. Real-time Clock and Count Down Clock

The operating system provides two different clocks, a continuously running real-time clock and a count down clock which can be started with a specified run time. There is only one real-time clock in common memory which is used by all Single Board Computers in the system. The real-time clock is driven by the Single Board Computer with the number 1. Both real-time clock and count down clock make use of

the interval timers on the Single Board Computers.

A count down clock is implemented in each Single Board Computer. The count down clock can be started at any time and generates an interrupt when the specified time is elapsed. The count down clock can be used to control time critical processes. It has a size of 16 bits where the least significant bit represents 1.86 micro seconds.

The real-time clock has a size of 4 bytes, the least significant bit has the value of 1 milli second and the maximum run time is approximately 50 days.

#### D. REAL-TIME EXECUTIVE

The real-time executive represents the kernel of the operating system. The executive continuously checks for pending tasks on four priority levels:

- 1. priority tasks
- 2. messages to be processed
- 3. periodic tasks
- 4. background tasks.

The processor is assigned to the next task with highest priority in this scheme. The executive only proceeds to the next lower level if no task is pending at the current or a higher level.

Each Single Board Computer runs its own, identical executive. An executive is tailored to its environment with special compile parameters.

The executives in different Single Board Computers communicate with each other using normal messages via an exchange in an absolutely located buffer in common memory.

Because the code of the executive is executed most often it is located in on-board memory.

#### E. INTERRUPT HANDLING

All interrupts are handled entirely by the operating system. The operating system initializes the interrupt controller and sets up the interrupt vector.

There are four special interrupts which are handled by the operating system: real-time clock interrupt, count down clock interrupt, system restart interrupt and an interrupt which causes the system to enter the monitor, if implemented.

User modules can activate interrupts with a system call and passing the requested interrupt level and the index of a priority task to process the interrupt. In case of an interrupt, a call of the priority task is scheduled by the operating system. The de-activation of an interrupt is also performed with a system call.

# F. SYSTEM MONITOR

The system monitor is implemented as a system call. It is called when internal program limits are exceeded. An address value and two byte values are passed with the system call. The address can point to the location of the error and the two byte values can further specify the nature of the error.

The system monitor generates the display of an appropriate message for the operator.

This feature is primarily designed as an aide in the program development and test phase and to cover undefined program states.

# G. SYSTEM INTEGRATION

User modules to run under the operating system are independent with respect to other user modules and the operating system. In order to provide the necessary links to the operating system, a user module needs to be compiled together with some system data and external declarations of the system calls. Furthermore the module's entry for the process of messages has to be identified to the operating system.

The linking of a module to the operating system is implemented using the public/external declaration feature of PL/M-80 and the LINK program in ISIS-II. Basically, a

system integration is the execution of this LINK program. Included in the linking process are the operating system itself and the user modules of the special configuration to be integrated.

Depending on the application, the linked and located code can be kept externally and loaded into Random Access Memory or transferred into Erasable and Programmable Read Only Memory.

# V. AVAILABLE USER MODULES

Three user modules have been developed and are available to run in the presented system configuration: a CRI module, a line printer module and a debug module. The software documentation, a program description of each module and a user's manual for the debug functions, is part of the Appendix.

# A. CRT MODULE

This module is a typical user module. It drives a CRI connected to a Single Board Computer. The CRT, via the CRI module, may be used by any other module in the system, regardless in which Single Board Computer it is located. Messages are used for the communication between the CRI module and a 'CRI user' module.

The CRT module provides two different kinds of outputs to the CRT and the facility of obtaining inputs from the connected keyboard.

# B. LINE PRINTER MODULE

The line printer module is the driver for a line printer. Any module in the system can transfer text with a message to the line printer module in order to have it printed.

### C. DEBUG MODULE

This module allows the debugging of the entire system under real-time conditions, i.e. without influencing system operation during the debugging process. The provided debug functions are designed to ease debugging of malfunctions which are typically encountered in real-time systems and specifically in Naval Tactical Data Systems.

The debugging concept allows several users to debug different program parts at the same time. Any Single Board Computer can be debugged from any other Single Board Computer with the restriction that only one user is allowed to debug a Single Board Computer at a time.

Input/output media for the debug module are a CRT or equivalent device and a high speed printer for output only. Both devices are driven by modules described in previous sections.

#### VI. CONCLUSIONS

In this section a comparison is made between the proposed computer system (hardware concept and operating system) and the requirements established in Chapter II.

It is emphasized that no attempt is made to examine the capabilities of the previously described computer system to drive a typical Naval Tactical Data System. Obviously the CPU, INTEL's 8080A, fails to qualify for this task because of its restricted instruction repertoire (especially the lack of arithmetic instructions), eight bit word size and 64K byte address space.

with the recent introduction of a single chip 16-bit microprocessor with

- instructions to operate on 8-,16- and 32 bit quantities
- signed and unsigned arithmetic operations including multiplications and divisions
- address space of 1 menabyte

the proposed model with minor adaptive changes in the operating system becomes realistic, provided that the concept of Single Board Computers will be kept. Considering the success of INIEL's SBC80 series, this is very likely.

# A. HARDWARE DESIGN

#### 1. Standardization and Cost

The proposed hardware design, distributed system with Single Board Computers on a common bus system, is very flexible. It can be used in Naval Tactical Data System applications which require extensive computing power as well as in very small and simple systems. Because of this flexibility, the proposed concept would reduce the number of different computer architectures currently in use. This reduction is equivalent to an increase in standardization which besides lower hardware cost has a strong influence on cost in terms of software development, maintenance and training of personnel.

#### 2. Expandability

The proposed concept has special advantages as far as future changes or extensions of the system are concerned. A hardware change is kept local to one or a few Single Board Computers. Extensions are accomplished easily by adding Single Board Computers to the system. However, it should be noted that the utilization of the system bus may turn out to be the 'bottleneck' of such a system. The capacity of the system bus clearly dictates the limits for the proposed computer design.

A solution to this problem is thinkable in form of a 'super bus' structure consisting of two or more of the system. This structure would lead to a strictly hierarchical system concept in which information is reduced before being transferred to the next higher bus level.

# 3. Interfaces with Peripheral Equipment

The implemented input/output interfaces in Single Board Computers are not fixed. The input/output and interrupt controller are software programmable and, in connection with easy to implement jumper connections, allow the tailoring of the input/output and interrupt facilities according to the application.

# 4. Maintenance and Reliability

The most astonishing effects of the new technology used in Single Board Computers can be found in the area of maintenance and reliability. A computer system consisting of Single Board Computers is practically maintenance-free. Although reliability tests of the new technology are still in progress, it is already known that there is a great increase in reliability. In case of a failure parts of the computer would no longer be replaced: the computer would be replaced itself.

# 5. Physical Requirements

The proposed computer system drastically reduces weight, space and environmental (power, cooling etc.)

requirements of currently implemented Naval Tactical Data Systems. This reduction is especially important for airborne systems.

#### 6. System Redundancy

Up to now a complete back-up computer system is not found in Naval Tactical Data System applications. Reasons for this are mainly costs and sometimes space and weight. With an implementation of the proposed system concept these constraints could be eliminated. Although reliability is already greatly increased a redundant system design can be considered.

#### B. OPERATING SYSTEM

#### 1. Naval Tactical Data System Requirements

The proposed operating system has been designed specifically for an application in Naval Tactical Data Systems. It provides facilities which ensure the real-time operation of the entire system. 'Real-time' is a relative expression with respect to Naval Tactical Data Systems. An operator expects a system reaction in real-time after completion of his inputs. In this case the system has to provide an 'instantenous' reaction in terms of human speed. However, in the case of a high frequency radar interface, 'instantenous' represents a considerably shorter time between input and reaction. The structure of the operating

system with different priority levels supports this interpretation of 'real-time' as well as other commonly found Naval Tactical Data System operations.

# 2. Software Development, Maintenance and Extensions

Because of the modular structure supported by the operating system, it is possible to run test vehicles early in the program development phase. The modules in the test vehicle are replaced by the original modules as soon as they are completed or the simulated equipment is installed. This concept of a continuous test of the growing system involves some simulation overhead but it avoids 'big bang' tests and leads to a better tested system.

The modular structure also eases program maintenance, because modules are easily changed and reintegrated into the system.

Extensions to the system are implemented by adding modules. Hardware facilities of the computer system can be extended by increasing the number of Single Board Computers. In order to find an optimal distribution of the extended program it may be necessary to re-distribute the modules over the Single Board Computers.

#### 3. Standardization

The structure of the operating system basically reflects a similar structure used in various real-time Naval

Tactical Data Systems. This allows the use of already developed software and knowledge.

#### 4. Simulation

Simulation in Naval Tactical Data Systems is needed mainly for three reasons: software development, software test and operator training. The modular software structure in connection with the distributed Single Board Computer concept allow simulation not only of missing modules or equipment, it allows the simulation of the operation of entire Single Board Computers as well. No hardware changes are necessary since the simulation takes place entirely inside the computer system.

# APPENDIX A

PROGRAM DESCRIPTION OF OPERATING SYSTEM

# TABLE OF CONTENTS

1	General	3
1.1	Introduction	3
1.2	Abbreviations and Conventions	3
2	Executive	4
2.1	Inputs	4
2.2	Function	4
2.2.1	System Initialization	4
2.2.2	Priority Calls	5
2.2.3	Communication between Modules	5
2.2.3.1	Internal	6
2.2.3.2	External	6
2.2.4	Periodic Calls	7
2.2.5	Background Tasks	7
2.2.6	Message Extraction	7
2.3	Outputs	8
3	Interrupt Handling	8
4	System Data	9
5	System Calls 1	0
6	Real-time Clock and Count Down Clock 1	1
7	Loader	2

8	Memory Map 13
8.1	Absolute Data
8.2	Absolute code
8.3	Changes of SBC Monitor
9 .	Listing of Executive Data 17

# 1 General

# 1.1 Introduction

This segment of text describes the function of the operating system. The use of the facilities is documented in the user's manual for the operating system.

In cases where system functions are well explained in the program listing itself no description is given here.

The executive is considered to be a module. It has the lowest possible module number in a SBC and the module identification Ex.

#### 1.2 Abbreviations and Conventions

All numbers in this segment of text are decimal except as otherwise indicated.

Task - part of the program which handles a specific function of the system and consists of one or more procedures

Module - part of the program which consists of one or more (related) tasks and can be compiled separately

Executive - part of the operating system which controls the scheduling and execution of tasks

Operating

System - consists of executive, system calls and system data

System - consists of operating system and all integrated user modules

SBC - Single Board Computer

MDS - Microcomputer Development System (INTEL)

MCB - Message Control Block

RMN - Receiving Module Number

SMN - Sending Module Number

MN - Message Number

ML - Message Length

EX - executive, module number in SBC 1/2/3: 00/08/16 LP - line printer module, module number in SBC 1/2/3: 05/13/21

CS - CRT module, module number in SBC 1/2/3: 06/14/22

DB - debug module, module number in SBC 1/2/3: 07/15/23

RTC - Real Time Clock

CDC - Count Down Clock

# 2 Executive

The executive is the kernel of the operating system. It controls the process of

- priority tasks
- real-time messages
- time dependent (periodic) tasks
- background tasks.

# 2.1 Inputs

EX receives two real-time messages from any debug module, 'start message extraction' and 'stop message extraction'.

#### Format:

RMN : EX

SMN : any debug module

MN : 10 - start

11 - stop

ML : 04

This message controls the state of the flag MSGEXTRACTION. It is set to 'IRUE' upon receipt of the 'start message extraction' message and reset to 'FALSE' when 'stop message extraction' is received.

#### 2.2 Function

#### 2.2.1 System Initialization

The system is initialized with a call of procedure EXSTART at system start. Prior to this call the variables SAVESTACKPTR and RESTART are set.

SAVESTACKPTR contains the value of the stack pointer at initial system start. It is saved for a system restart without loading and system RESET.

RESIARI is set to 'FALSE' at the initial start of the system. In case a system restart is initiated with INT6, RESTART is set to 'TRUE'. At the same time the stack pointer is reset to the saved value.

RESIART is used by all modules to determine whether the current start is a system start with or without hardware reset. In order to prevent overlapping program action caused by erroneous interrupts, the interrupts are locked out for the time of system initialization.

All system tables are reset and a 'start' message for each module in the same SBC is packed into the system's message buffer.

EXSTART ends with the initialization of the interrupt controller and an ENABLE instruction.

SRC1 has additional tasks at system initialization. It resets the RIC, starts the RIC update and gives the start signal to all other SBCs in the system.

All other modules in the system check their specific start variable START1, START2 etc. to take on a value other than 0. After completion of the initialization, SBC1 sets the respective SBC number into START1, START2 etc. and all SBCs start their initialization.

#### 2.2.2 Priority Calls

In the context of priority calls there are two relevant items of system data: PRIORLIST and PRIORSCHEDULE.

PRIORLIST is an address vector of length 8 and contains the addresses of priority procedures entered.

PRIORSCHEDULE is a byte variable which indicates the scheduled priority calls, e.g. bit 0 = 1 means that a priority call of the priority procedure in PRIORLIST(0) has been scheduled. Prior to the call of this procedure the respective bit in PRIORSCHEDULE is reset to 0.

The executive keeps checking PRIORSCHEDULE until all calls have been made before proceeding to the process of real-time messages.

#### 2.2.3 Communication Between Modules

Communication between modules uses real-time messages. These messages can be sent from any module to any other module in the system.

A message is considered to be 'internal' if it is sent to a module in the same SBC. An 'external' message is sent to a module in an other SBC.

Internal and external messages have the same format, only the treatment by the executive is different.

A message is sent with a call of SEND. In this system procedure the message is placed into MSGBUFFER, a circular FIFO list.

MSGBUFFER is controlled by two pointers: MSGIN (next to fill) and MSGOUF (next to process). The variable NUMMSG contains the number of messages currently in MSGBUFFER.

#### 2.2.3.1 Internal

The executive checks NUMMSG for a message to be processed. If NUMMSG = 0 then the executive proceeds to check for external messages.

MSGOUT points to the next message to be processed. The executive computes the index of the next message in MSGBUFFER (new MSGOUT = MSGOUT + ML of current message). After this update, the executive examines RMN of the message. If the receiving module is in the same SBC the procedure MSGENTxx is called (xx = relative module number in a SBC : 0 = 7). This procedure is the message entry of the receiving module.

# 2.2.3.2 External

If the receiving module of a message is not in the own SBC, the executive calls SENDEXT to process this message.

There is a buffer for external messages (EXTMSGBUFFER) which has a very similar structure to MSGBUFFER.

The only exception is that the receiver of the message is the number of the SBC which hosts the receiving module.

An external message is kept into EXTMSGBUFFER until it is processed by the respective SBC and transferred into the local message buffer.

Since all SBCs operate in EXIMSGBUFFER there is a lock mechanism that prevents two SBCs from working in EXIMSGBUFFER at the same time.

Every time a message is taken out of EXTMSGBUFFER or when the first message is written into the empty EXTMSGBUFFER the variable EXTMSG is set to the number of the receiving SBC of the message currently at the top of EXTMSGBUFFER.

If EXTMSG = 0 then no external message is waiting to be processed.

After checking the external messages PRIORSCHEDULE is examined again.

#### 2.2.4 Periodic Calls

All activated periodic calls are kept in PERLIST.
PERLIST is a vector of records.
The variable NUMPER contains the number of activated periodic calls.

PERXIBL, a list of pointers to PERLIST, is always kept compact, i.e. if a periodic call is suspended, entries in PERXIBL are moved to become compact again. This technique reduces execution time when the executive is checking for necessary periodic calls.

If the executive finds a 'next call time' <= RIC then a new 'next call time' is computed (RIC + time interval) and the periodic procedure is called.

If no periodic procedure is to be called, the executive proceeds to the background tasks. Otherwise the periodic procedure is called and upon return of program control the priority calls are checked again.

# 2.2.5 Background Tasks

Background tasks represent the lowest priority level within the executive. They are executed only if no other task is pending, i.e. the processor is idling.

#### 2.2.6 Message Extraction

Before processing a real-time message, the executive calls EXMSGEXTR if the flag MSGEXTRACTION is 'TRUE'. In this procedure the current message is checked against the message control block contained in DEBUGMCB. If DEBUGMCB matches the current message, the message entry of the debug module is called with a faked 'message extraction' message.

Upon return the current message is processed.

#### 2.3 Outputs

At system start, after its own initialization, EX sends 'start' messages to all modules in the same SBC.

## Format:

RMN - all modules in own SBC

SMN - EX

MN - 00

ML - 04

Apart from the 'start' message, EX sends an 'extraction' message to DB if message extraction is activated and a matching message was detected. This message is 'sent' by directly calling the message entry of DB.

This special procedure is chosen in order to avoid an excessive load of the system's message buffer since each extracted message would be represented twice: as original message and as data bytes of the 'extraction' message.

#### 3 Interrupt Handling

All interrupts are handled entirely by the operating system.

There are four system interrupts and three user interrupts. The system interrupts are:

- RIC interrupt
- CDC interrupt
- system restart
- enter monitor.

The RTC interrupt (INI2) is activated in SBC 1 only. This interrupt is generated by counter 0 of the interval timer arriving at the terminal count of 0. Counter 0 is first set in procedure EXSTART to the equivalent of 1 msec and started. Upon occurrence of the interrupt the RTC is updated and the counter is started again with a time interval of 1 msec.

A CDC interrupt may be initiated in each SBC. The CDC interrupt is started with the system call SETCDC.

At terminal count the interrupt (INTO) is generated and the address passed with the system call is called.

The system can be restarted without RESEI by generating INT6 at the front panel of the MDS. From the interrupt process the procedure SYSRESTART is called where the restart is initiated.

The SBC monitor can be entered at any time by pressing INT2 at the front panel.

This interrupt is jumpered on the SBCs to cause an interrupt on level 1.

Upon occurrence, the monitor is entered at location 0740H. The same procedure when activating the monitor with RESET applies, i.e. typing capital 'U' to initialize the USARI.

Note: Since the monitor changes locations in on-board Random Access Memory, the system cannot be restarted without loading!

User interrupts can be activated on levels 3,4 and 5. They are activated and de-activated with system calls (ENTERINT and REMINT).

The interrupt vector is located at 3000H and has a length of 64 bytes, i.e. each interrupt occupies 8 bytes.

This structure is compatible with PL/M-80.

The interrupt routines are written in PL/M-80 and therefore located at 0000 - 003FH.

After SBC 1 is loaded, the loader transfers the code for interrupt processing to its final location in the interrupt vector.

Since the user interrupts are restricted by PL/M-80 to INT3 - INT7, only 5 interrupts can be programmed this way. These are the three user interrupts and RTC and CDC interrupt. The code for the process of monitor and restart interrupt is written into the interrupt vector in procedure EXSTART.

The book-keeping of user activated interrupts takes place in table INTTBL.

INITBL(i) contains FFH if interrupt i is not activated. An activated interrupt is indicated by INITBL(j) = k, where j is the interrupt level and k is the index of the priority task to be scheduled upon occurrence of the interrupt.

System Data

4

System data are divided in two parts:

- data to be compiled with each module
- data to be compiled with the executive, system calls and the debug module.

The first data set is in the source files SYDATP.SRC and SYDATE.SRC.

The executive has to be compiled with the PUBLIC declarations of this data in SYDATP.SRC whereas all other components of the system are compiled with the EXTERNAL declarations in SYDATE.SRC.

Since this data set is well explained in the program listing (see Section 11 in the operating system user's manual) it is not described here.

The second data set is in the source files EXDATP.SRC and EXDATE.SRC.

It contains all data necessary for the operation of the executive and the system calls.

The executive has to be compiled with the PUBLIC declarations in EXDAIP.SRC. All other components which need to operate on these data (system calls, interrupt handling, debug module) can be compiled with the EXTERNAL declarations in EXDATE.SRC.

All operational data of the system are listed and described in Section 9.

### System Calls

5

The code of the system calls has been split up into seven parts in order to ease program development and maintenance under ISIS-II. These program parts are named SCPUB1 - SCPUB7.

The object code of the system calls is kept in the object library SC.LIB.

Each module can be compiled with the set of EXTERNAL declarations of all system calls in the source file SCEXT.SRC. The matching of the PUBLIC and EXTERNAL declarations takes place in the LINK step during system generation where the object library SC.LIB has to be included.

The function of the system calls is explained in the source program listing.

Real-time Clock and Count Down Clock

6

RIC and CDC are implemented using counter 0 and 1 of the on-board interval timer.

Both counters are 'down counters' with a terminal count of 0 and driven by a clock input of 1.86 micro seconds.

The 'terminal count' output line is jumpered to the interrupt controller.

Counter 0 generates a level 2 interrupt while counter 1 is tied to INIO.

Counter 0 (RTC) is driven by SBC 1 only. SBC 1 loads counter 0 with the equivalent of 1 msec (LSB = 1.86 micro seconds) and updates the RTC (4 byte vector in common memory) by 1 upon occurrence of the interrupt, i.e.terminal count of counter 0.

A CDC interrupt is implemented in each SBC. Its initialization is by a system call (SETCDC). / In the process of this system call the following actions take place:

- save the address to be called at CDC interrupt
- enable interrupt level 0
- load counter 1 with the transferred value (LSB = 1.86 micro seconds).

Upon occurrence of the CDC interrupt, the interrupt on level 0 is disabled and the indicated address is called.

Loader

7

The system is loaded and started under control of a separate loader.

The loader runs in the MDS and is started together with the SBCs. It interacts with the SBCs through four absolutely located variables:

- LOADSBC (F1F0H)
- STARTI (FIFIH)
- START2 (F1F2H)
- STARTS (F1F3H).

At start all four variables are reset to 0. The SBCs continuously check LOADSBC to take on the value of their SBC number (1-3).

The loader loads the code for SBC1 from the disk with an offset(bias) of 6000H.An ISIS-II system call is used to load the file LUAD1.

After completion of the loading LOADSBC is set to 1. The loader now waits until LOADSBC becomes 0 again.

SBC1 detects the '1' in LOADSBC and starts moving the code from the temporary storage into on-board memory for which it was located before by the LOCATE program of ISIS-II.

After completion of the move, LOADSBC is set to 0 and then the SBC waits for the variable START1 to become 1.

The loader now repeats the process for SBC2 and SBC3.

After having loaded all SBCs, the loader stores a 1 in START1. This is the signal for SBC1 to start. After initializing system data and the RTC the variables START2 and START3 are set to 2 and 3 respectively. This effects the start of the entire system.

The loader issues informative messages on the CRI as it proceeds through the loading process.

It should be noted that a loading process is not necessary in a practical application because the program would reside in Read Only Memory.

#### B Memory Map

In this section all absolutely located data and code is described. Furthermore all PRUM changes of the SBC monitor are listed.

## 8.1 Absolute Data

Absolutely located data is necessary for communication between SBCs and loading and starting of the SBCs.

Absolute system data are located in the region F000H - F1EFH. These data include:

- module status table (MODSTATUS)
- real-time clock (RIC)
- message buffer and message control variables for external message exchange.

F000	
	MODSTATUS
F017	
F018	
	RTC
F018	
FOIC	EXIMSGLOCK
FOID	EXTMSG
FOIE	EXTMSGIN
FOIF	EXIMSGOUT
F020	NUMEXIMSG
F021	EXTLASIMSG
F022	
	EXTMSGBUFFER
F119	

Absolute data for loading and starting are located in the region F1F0 - F1F7:

FIFO	LOADSBC
FIFI	STARTI
F1F2	START2
F1F3	STARTS
F1F4	
• •	key for the system operation (key is OABCDH)
F1F5	

Since the snapshot function is implemented with the trap instruction RSI4, it is necessary to define the entrance of the snapshot execution in an absolute location. The entry address is stored in 3040H and 3041H.

### 8.2 Absolute Code

Apart from the code for the SBC monitor, which is located absolutely because it resides in ROM, the interrupt vector has to be located absolutely.

The interrupt controller is programmed to expect the interrupt vector at 3000H with 8 bytes/interrupt. The code for INTU, INT2, INT3, INT4 and INT5 is moved into the vector by the loader. The rest of the vector is set in the start routine (EXSIARI). These are INT1 (entry of monitor) and INT6 (system restart). INT7 currently is not implemented.

3000 CDC interrupt 3007 3008 'enter SBC monitor' interrupt 300F 3010 RIC interrupt 3017 3018 INI3 (user interrupt) SOIF 3020 INI4 (user interrupt) 3027 3028 INTS (user interrupt) 302F 3030 'system restart' interrupt 3037 3038 IN17 (not implemented) 303F

# 8.3 Changes of the SBC Monitor

This section lists and comments the changes of the SBC monitor in PROM.

The monitor needs to be modified in order to allow the automatic loading of the system. At system start the monitor checks a key which is an address value located in F1F4H. For operation of operating system the user has to store the the hexadecimal value ABCD in this location. If the contents is different from this value the monitor will operate in normal mode.

0000	JMP 0700	C3 00	07 jump to 0700H at RESET
0700	LXI H,KEY	21 F4 F1	check key
0703	MVI A, OABH	SE AB	
0705	CMP M	BE	
0706	JNZ 0740	C2 40 07	start monitor
0709	INX H	23	
070A	MVI A.OCDH	SE CO	
070C	CMP M	BE	
0700	JNZ 0740	C2 40 07	start monitor
0710	10	F3	disable interrupts
0711	LXI H, LUADSBC	21 FO F1	wait until ready to load
0714	MVI A.SBC	3E nn	SBC nn
0716	CMP M	BE	
071/	JNZ 0716	C2 16 07	not ready yet
UTIA	LXI B,9000	01 00 90	begin of temporary code
0710	LXI 0,3042	11 42 30	begin on-board RAM
0720	LDAX B	OA	
0721	STAX B	15	move code into own RAM
0722	INX B	03	
0723	INX D	13	
0724	MVI A, OEFH	3E EF	last byte to be moved
0726	CMP C	89	
0727	JNC 0720	DS 50 07	continue move
072A	MVI A,0	3E 00	
0720	STA LOADSBC	32 F0 F1	reset LOADSBC
072F	LXI H, STARTH	21 Fn F1	check for start SBC n
0732	MVI A.O	3E 00	
0734	CMP M	BE	
0735	JZ 0734	CA 34 07	no start yet
0738	STA STARTA	32 Fn F1	reset STARIn
0738	JMP 3042	C3 42 30	start SBC
0740	MVI A.4E	3E 4E	replaced monitor
0742	OUT ED	D3 ED	instructions
0744	JMP INUST	C3 DH U3	0000 - 0006

0050	DI		F3	execute snapshot
1500	PUSH	H	E5	upon execution of
2500	PUSH	D	05	a RST4 instruction
0023	PUSH	В	C5	
0024	PUSH	PSW	F5	save registers
0025	LHLD	3040H	2A 40 30	address of snapshot
8500	PCHL		E9	transfer control

Listing of Executive Data

```
MODULE AUTHORIZED TO SUSPEND/ACTIVATE
                                                                                                                                               MODSTATUS CONTAINS THE STATUS OF ALL MODULES IN THE SYSTEM. INDEX IS MODULE NUMBER.
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       MODULE NOT AUTHORIZED TO SUSPENDA
                                                                                                                                                                                                                                                                                                                                   MODSTATUS IS UPDATED BY EXEC WITH SYS MSG 4 (BIT 0 = LSB)
                                                                                                                                                                                                                                                   DCL MODSTATUS (MAXSYSMOD) BYTE PUBLIC AT (BEGINCM);
                                                                                                                                                                                                                                                                                                                                                                          BIT 0 - BIT 7 = 0: MODULE DOES NOT EXIST
BIT 0: 1 - MODULE EXISTS
                                                                                                                                                                                                                                                                                                                                                                                                                                                             MAY NOT BE SUSPENDED
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 MODULE MAY BE SUSPENDED
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  (MODSTAT BASED ADRSTAT) (MAXMOD) BYTE;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           ACTIVATE OTHER MODULES
                                                                                                                                                                                                                                                                                                                                                                                                                    0 - MODULE NOT ACTIVATED
                                                                                                                                                                                                                                                                                                                                                                                                                                          MODULE ACTIVATED
                                                                                                                                                                                       EXECUTIVE DATA
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      THER MODULES
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            ADRSTAT ADDRESS PUBLIC,
                                                                                                                                                                                                                                                                                                                                                                                                                                                               MODULE
                    # INCLUDE(:F1:EXDATP, SRC)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        Ø
                                                                                                                                                                                                                                                                                                                                                                                                                                                                 Ø
                                                            UPDATE:
                                                                                                                                                                                                                                                                                                                                                                                                                      BIT 1:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        BIT 3:
                                                                                                                                                                                                                                                                                                                                                                                                                                                               BIT 2:
$EJECT
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               ಕ್ಷ
                                                                                                                                                                                                                                                    16
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               11
```

18 -

60

ທ

12

9

# PL/M-80 COMPILER

<pre>= EXTLASTMSG BYTE PUBLIC AT (. BEGABSDATA + 9), = EXTMSGBUFFER (250) BYTE PUBLIC AT (. BEGABSDATA + 10),</pre>	/* EXTERNAL MESSAGE BUFFER		ENDABSDATA BYTE PUBLIC AT (. BEGABSDATA + 258);		END OF ABSOLUTE DATA IN COMMON MEMORY		DCL (LOADSBC, START1, START2, START3) BYTE PUBLIC AT (0F1F0H);		RESOLUTE LOCATED VARIABLES USED FOR LOADING AND		*	DCL ILLMSG (36) BY			TEXT FOR CRT MSG / ILLEGAL MSG/		מכר	MONTEXT (26) BYTE PUBLIC INITIAL ('SYSTEM MONITOR: 8888 88 88'	~ ·	*	SYSTEM MONITOR MSG TO CS		DCL SAVESTACKPTR ADDRESS PUBLIC;	*	STACK POINTER AT SYSTEM START		DCL EXTRMSG (4) BYTE PUBLIC INITIAL (DB, EX, 27, 0);		EXTRACTION MSG TO DEBUG MODULE
11 11	11 11	11	11	11	11	11.	H	11	11	11	11	11	11	11	II	11	11	11		11	11	11		11	11	11	11	11	11
							4					4					स						7				4		
							13					14					15						16				17		

# PL/M-80 COMPILER

# Not choose Annersa Pile To:		= ADDRESS TO BE CALLED AT CDC INTERRUPT	*	DCL RSTADR ADDRESS PUBLIC, RSTFL BYTE PUBLIC;	*	= ADDRESS OF PROCEDURE SYSRESTART	*	<ul> <li>DCL SNAPINTADR ADDRESS PUBLIC AT (0F1F6H);</li> </ul>	*	- RDDRESS OF SNAPSHOT INTERRUPT ENTRY	*	= DCL EXCLEARBEG BYTE PUBLIC;	*	- BEGIN OF DATA TO BE CLEARED AT START	*	= DCL INTTBL(8) BYTE PUBLIC;	**	= TABLE CONTAINS INDICES OF PRIORITY PROCEDURES TO	■ BE SCHEDULED IN CASE OF AN INTERRUPT	·*	≈ DCL CDCACTIVE BYTE PUBLIC:	= /* TRUE IF CDC INT ACTIVATED */	■ DUL CODESAVE ADDRESS PUBLIC,	*	≈ SYSTEM MONITOR CODE FOR MSG BUFFER OVERFLOW	\*	= EXPERFL BYTE PUBLIC,	≈ EXPERADE ADDRESS PUBLIC,	= EXPERX BYTE PUBLIC:	
•				4				4				4				₹					त		त							
	1			13				58				77				22					23		24							

= DCL (XB1, XB2, XB3) BYTE PUBLIC, (XA1, XA2, XA3) ADDRESS PUBLIC; = /*	EXEC WORK VARIABLES	DCL PERXTBL (MAXPER) BYTE PUBLIC;	*	PERXTBL CONTRINS POINTER TO PERLIST IN COMPACT FORM		* DCL PERLIST (MAXPER) STRUCTURE (	PERHOR HODRESS,	PERINTADE RODRESS, /* ADDRESS OF TIME INTERVAL */	PERTIME (4) BYTE) F	PERLIST CONTRINS ALL SIGNIFICANT DATA OF AN		*	: (PERINT BASED XA1) (4) BYTE,	*	OVERLAY FOR TIME INTERVAL OF A PERIODIC	*	PERX BYTE PUBLIC,	*	PERX IS SEARCH INDEX FOR PERLIST	*	NUMPER BYTE PUBLIC;	*	MUMBER OF ACTIVATED PERIODICS	·	DCL EXCLERREND BYTE PUBLIC:
₩.		" ਜ				H		•		"	"	"		"	"		"			"	"			"	4
28		53				36																			31

\*

PAGE 10

PL/M-80 COMPILER

END OF DATA TO BE CLEARED AT START \*/

# APPENDIX B

USER'S MANUAL FOR OPERATING SYSTEM

# TABLE OF CONTENTS

1	Genera	1	• • • •	• • • •	•••			• • •	• • •	• • • •		•••		•••	••	3
1.1	Introd															3
1.2	Abbrev															3
1.3	Facili	ties	of	the	Ope	rat	ing	Sy	ster	n	•••	•••	• • •	•••	• •	4
2	System	0.00														5
2.1	Modula															5
2.5	Priori															6
2.3	Commun															7
2.4	Time D		A				1000	AND RESERVE		TO 1					700	8
2.5	Backgr	ound	Tas	ks	•••	•••	•••	•••	•••	• • • •	• • • •	•••	• • •	•••	••	8
3	Intern	tau	Hand	ling						• • •						8
4	System	Dat	a	• • • •	•••	•••	•••	•••	•••	• • • •	• • • •	••	• • •	•••	••	9
5	Execut	ive.			•••	•••		•••	•••	•••	••••	••	•••	•••	••	9
6	System	Cal	15			•••		•••	• • •	• • • •		••			••	10
6.1	ENTERP															10
6.2	PRIORI															10
6.3	REMPRI															11
6.4	SEND															11
6.5	PERACT														-	11
6.6	PERCHG	-														12
6.7	PERSUS															12
6.8	ACTIVA															12
6.9	ACTIVE					-										12
6.10	SUSPEN														-	13
6.11	PROCAD															13
6.12	UPDSTA															14
6.13	CLEARD															14
6.14	811															14
6.15	CLEARH															14
6.16	SETBIT															15
6.17	SETCOC															15
6.18	ILLEGA															15
6.19	ENTERI															15
6.20	REMINT	••••	• • • •	• • • •	•••	•••	•••	•••	•••	•••	••••	•••	• • •	•••	• •	15
7	System	Mon	itor												!	16

8 9 10 11	Real-time Clock and Count Down Clock	
	Listing of System Data	3
	Listing of External System Call Declarations 2	7

#### General

#### .1 Introduction

This manual describes the use of the real-time operating system for dedicated multi processor micro computers. The operating system is designed to use INTEL's single board computers SBC80/20-4, therefore it is assumed that the reader is familiar with

- single board computer hardware
- PL/M-80
- operating system ISIS-II for INTEL's MDS.

# 1.2 Abbreviations and Conventions

All numbers in this segment of text are decimal except as otherwise indicated.

Task - part of the program which handles a specific function of the system and consists of one or more procedures

Module - part of the program which consists of one or more (related) tasks and can be compiled separately

Executive - part of the operating system which contols the scheduling and execution of tasks

Operating

System - consists of executive, system calls and system data

System - consists of operating system and all integrated user modules

SBC - Single Board Computer

MDS - Microcomputer Development System (INIEL)

MCB - Message Control Block RMN - Receiving Module Number SMN - Sending Module Number

MN - Message Number
ML - Message Length

EX - executive, module number in SBC 1/2/3: 00/08/16 LP - line printer module, module number in SBC 1/2/3: 05/13/21 CS - CRI module, module number in SBC 1/2/3: 06/14/22 DB - debug module, module number in SBC 1/2/3: 07/15/23

RTC - Real Time Clock CDC - Count Down Clock

### 1.3 Facilities of the Uperating System

The operating system is specifically designed to control complex real-time environments. In order to meet these requirements the operating system provides:

- priority task scheduling
- communication between modules
- time dependent (periodic) scheduling of tasks
- real-time clock and count down clock
- system monitor
- commonly needed functions in form of system calls
- interrupt handling.

The operating system is able to control all possible SBC hard-ware configurations. INTEL's multibus system can handle up to 1b master controller, theoretically all SBCs.

Code in memory may be shared by several SBCs, however, care should be taken that there is no interference with data access. By simply keeping all data in on-board memory no data conflicts can occur.

For efficiency reasons the executive and time critical functions should be kept in on-board memory, whereas the code of the system calls can be located in common memory where it can be shared.

A module can take on the identities of several modules. The body of the module is shared in common memory. The kernel of the module, i.e. the entry for real-time message, is special in each SBC as expressed by different module numbers.

Communication between modules allow the sending of messages from a module to another module, regardless of where these modules are located.

Since message between SBCs as well as code executed from common memory make use of the system bus the actual distribution of the modules in the entire computer system should be such that system bus access is minimized.

All executives in the SBCs are identical, the binding of an executive to the host SBC takes place during the compile with two compile paremeters: number of SBC and module number of first module in the SBC.

System Organization

1.1 Modular Structure

5

The operating system supports a strictly modular structure. Modules are integral parts of the overall system which usually handle a specific task or a group of related tasks. This organization not only eases program development and maintenance it also allows easy-to-implement changes of the system.

The links between a module and the rest of the system are the system data and the entry for real-time messages of the module.

These links require

- the inclusion of system data in the compilation of a module
- the marking of the message entry procedure as a PUBLIC procedure and a predefined name to allow access by the executive.

The example of a complete user module is given in Section 9.

A module has a unique number which is used to identify it in the system.

The number of a module depends on the number of the SBC which hosts the module. Each SBC can have a maximum of 8 modules (0-7), e.g. module 3 in SBC 2 has the module number 11. The lowest module number in each SBC is reserved for the executive. Implementing the executive as a module allows user modules to send message to the executive.

Each possible module in the system has a reserved byte in the system table MODSTATUS. This table contains the status of all modules. If the entry is 0 for a module the module is not existent.

MODSTATUS is updated with the system call UPDSTAT (see Section 6.12).

Each module is in one of three states: nonexistent, existent or activated.

### 2.2 Priority Tasks

Priority tasks consists of one or more procedures with one entry procedure. The call of this procedure can be scheduled. A scheduled priority task will be considered by the executive as soon as the processor becomes available.

Usually the process of an interrupt is implemented as a priority task where the scheduling is done in the interrupt procedure.

In connection with priority tasks there are four related system calls: PRIURITY, PRIORITYINT, ENTERPRIOR and REMPRIOR (see Section 6 for formats).

A priority task has to be entered into the list of priority calls prior to its first scheduling. This is done with the system call ENTERPRIOR. ENTERPRIOR returns an index which is used to schedule a call of the task (PRIORITY or PRIORITYINI) or remove the task from the priority list (REMPRIOR).

PRIORITYINI and PRIORITY perform the same function, however, PRIORITY may not be called from an interrupt procedure and vice versa. This prevents erroneous program action in the case that the processor is interrupted while executing the priority scheduling system call.

### 2.3 Communication Between Modules

Since modules are separate and independent parts of the system, the operating system has to provide a function which allows the modules to communicate with each other. This communication takes place in form of a message exchange. Messages are sent and received by modules and the sender and receiver are identified by their module numbers.

A message consists of a message control block (MCB) and, if necessary, data bytes.

The MCB has a length of 4 bytes and the following structure:

\*\*\*\*\*\*\*

\* RMN \* Receiving Module Number

\*\*\*\*\*\*\*

\* SMN \* Sending Module Number

\*\*\*\*\*\*\*

\* MN \* Message Number

\*\*\*\*\*\*\*\*

\* ML \* Message Length

The MN is an agreed upon number between sender and receiver of the message in order to identify the meaning of the message. ML determines the total number of bytes of the message (MCB + data bytes) and has a minimum value of 4 (message without data bytes).

A message is sent with the system call SEND. This routine requires the address of the first byte of the MCB to be passed to it. SEND returns a 'TRUE' if the message has been sent and a 'FALSE' otherwise. The latter can happen when the receiving module is not activated or not existent.

If the message has been sent, data bytes may be stored in the vector MSGDATA. SEND has reserved space for as many data bytes as indicated by ML in the MCB.

In the case that more data bytes than specified are written into MSGDATA they will be disregarded.

The message is inserted into a circular FIFO list.

The executive checks the top of the list and transfers control to the message entry of the module specified by RMN in the MCB. Prior to calling the module the message to be processed is set into the vectors MSG and MSGDAIA to allow access by the processing module.

### 2.4 Time Dependent (Periodic) Scheduling of Tasks

A common requirement of real-time applications is the time dependent execution of tasks, e.g. a display has to be updated every two seconds or a process has to be triggered once after a certain amount of time has elapsed. The operating system provides the functions for executing tasks of this kind.

The time dependent scheduling of tasks is implemented with three system calls: PERACT, PERCHG and PERSUSP (see Section 6 for formats).

A procedure can be entered into the list of periodic tasks by calling PERACT and passing 2 parameters: the address of the procedure entry and the time interval. PERACT returns an index which identifies the task in the system calls PERCHG and PERSUSP.

The time interval can be changed with the system call PERCHG and a periodic task can be removed from the list with a call of PERSUSP.

The executive continously checks the list of periodic tasks and calls the respective procedure when the specified time is less than or equal to the value in the real-time clock.

### .5 Background Tasks

In order to utilize the processor in case that no priority call is scheduled, no message is to be to processed and no periodic call is necessary, background tasks can be called by the executive.

Typically, not time dependent hardware checks are performed as a background tasks on a time slice basis.

### 3 Interrupt Handling

All interrupts are handled by the operating system. User interrupts currently can be activated on three levels: INT3, INT4 and INT5.

An interrupt is activated with the system call ENTERINT. Parameters for this system call are the requested interrupt level and the index of the priority task to be scheduled upon occurrence of the interrupt.

An activated interrupt can be de-activated with the system call REMINT.

### System Data

The set of EXTERNAL system data declarations as listed in Section 11 has to be included in the compilation of a module. The PUBLIC definitions of the same data are compiled together with the executive.

Care should be taken in the use of the system work variables. Since all modules are allowed to use them (except in interrupt routines) their contents is not preserved from one call to the next.

### Executive

This section describes the function of the executive and the interface between executive and user module.

The executive of the operating system is a program loop which checks for pending tasks on 4 levels:

- priority tasks
- messages to be processed
- periodic tasks
- background tasks.

The executive only proceeds to the next lower level if no task is pending at the current or a higher level.

The only link between the executive and a user module is the message entry of the module.

The executive is compiled with all possible message entries declared as EXTERNAL procedures while the respective PUBLIC declaration is part of the module.

During the linking process of ISIS-II the two declarations are matched and the ececutive can call the message entry of the module at system start. It is up to the module to initialize itself, enter priority calls, activate periodic calls and set its own status.

Prior to calling a module's message entry, the executive 'sets' the message into the based vectors MSG and MSGDATA where MSG contains the MCB and MSGDATA the data bytes, if any.

Each SBC runs its own, identical executive. The only difference between the ececutives is the module number.

### 6 System Calls

This section describes format and function of the system calls.

All system calls are given in the form

NAME TYPE (parameter list).

A TYPE included after the name means that the system call returns a value of the indicated type.

An/Bn in the parameter list indicate address/byte values.

### 6.1 ENTERPRIOR

Format : ENTERPRIOR BYTE (A1)

Input : A1 - address of priority procedure

Output : . index of the priority task

(may be used for scheduling and removing the task)

 FF if the task could not be entered into the list of priority tasks (overflow)

Function: The indicated priority procedure is entered into the list of priority tasks. The index to this list is returned. The list can hold up to 8 tasks/SBC.

### 6.2 PRIORITYINT/PRIORITY

Format : PRIORITYINT/PRIORITY (81)

Input : B1 - index of priority task

Function: A single call of the indicated priority procedure is initiated by the executive. The call occurs as soon as the CPU becomes available.

Remarks: PRIORITYINI is to be called from interrupt procedures only and PRIORITY is not to be called from interrupt procedures!

Prior to scheduling a priority task it has to be entered with ENTERPRIOR, otherwise erroneous program action will occur.

### 6.3 REMPRIOR

Format : REMPRIOR (81)

Input : 81 - index of priority task

Function: The indicated priority task is removed from the list of priority tasks and can no longer be

scheduled.

### 6.4 SEND

Format : SEND BYTE (A1)

Input : Al - address of first byte of MCB

Output : TRUE if message sent

FALSE otherwise

MSGDATA is set to contain the data bytes of the

message.

Function: A message is sent to another module with this

system call.

If the output is IRUE the message is sent and the data bytes can be written into MSGDATA, if any.

If the output is FALSE the message could not be sent.

This may be caused by a not activated receiving module or an overflow in the system's message buffer.

### 6.5 PERACT

Format : PERACT BYTE (A1, A2)

Input: Al - address of entry procedure for periodic task

A2 - address of first byte of time interval (The time interval is expected to be in RTC format.)

Output : . index of periodic task

. FF if task could not be activated because of an

overflow in the list

Function: The call of the indicated periodic task is activa-

ted with the input time interval.

Remark: The task is called as soon as possible and from

then on with the time given interval.

### 6.6 PERCHG

Format : PERCHG (B1,A1)

Input : B1 - index of periodic task

A1 - address of first byte of time interval (in RTC

format)

Function: The time interval of the periodic task is changed.

Remark : The periodic task is called next at RTC + new

time interval.

### 6.7 PERSUSP

Format : PERSUSP (B1)

Input : B1 - index of periodic task

Function: The periodic task is removed from the list of per-

iodic tasks and is not called any more.

### 6.8 ACTIVATE

Format : ACTIVATE BYTE (B1, B2)

Input : B1 - number of module to be activated

B2 - number of calling module

Output : FALSE if module to be activated does not exist or

calling module not authorized

TRUE otherwise

Function: The indicated module is to be activated.

The module receives a 'restart' message to indicate

that it has to reinitialize itself.

### 6.9 ACTIVE

Format : ACTIVE BYTE (B1)

Input : 81 - module number

Output : TRUE if the indicated module is active

FALSE otherwise

Function: The system call checks the status of the input

module and returns a TRUE if the module is active.

### 6.10 SUSPEND

Format : SUSPEND BYTE (81,82)

Input : B1 - number of module to be suspended

B2 - number of calling module

Output : FALSE if the module cannot be suspended or calling

module not authorized

TRUE otherwise

Function: The indicated module receives a 'suspend' message.

Its status is changed from 'active' to 'existent'.

### 6.11 PROCADR

Format : PROCADR ADDRESS

Output : first address of calling procedure

Function: This system call is used to determine the location

of a procedure at run time.

Remark : Since PROCADR examines the system stack, it only

returns a correct value if called with the

following sequence of code:

XYZ: PROCEDURE;

IF FLAG = 0 THEN

00;

XYZADR = PROCADR;

FLAG = 1; RETURN;

END;

. .

procedure body

END XYZ;

### 6.12 UPDSTAT

Format : UPDSTAT (B1,82)

Input : B1 - module number

B2 - status

bit 0 : 0 - module does not exist

1 - module exists

bit 1: 0 - module not acticated

1 - module activated

bit 2: 0 - module may not be suspended

1 - module may be suspended

bit 3: 0 - module not authorized to suspend/activate others

1 - module authorized to suspend/

activate others

Function: The current status of module B1 is replaced by B2.

Remark: The status of a module has to be set when the module receives the 'start' message in order to change its status from 'not existent' to 'exis-

tent' or 'activated'.

### 6.13 CLEARDATA

Format : CLEARDATA (A1,A2)

Input : Al - address of first byte

A2 - address of last byte

Function: Clear data from A1 to A2.

### 6.14 BIT

Format : BIT BYTE (81,82)

Input : B1 - number of bit

82 - byte to be checked

Output : TRUE if bit 81 in 82 is set

FALSE otherwise

### 6.15 CLEARBIT

Format : CLEARBIT (B1, A1)

Input : B1 - number of bit

A1 - address of a memory location

Function: Bit B1 in memory location A1 is set to 0.

### 6.16 SETBIT

Format : SEIBIT (B1, A1)

Input : B1 - number of bit

A1 - address of memory location

Function: Bit B1 in memory location A1 is set to 1.

### 6.17 SEICDC

Format : SETCDC BYTE (A1, A2)

Input : A1 - time interval (LSB = 1.86 micro sec)
A2 - address to be called at CDC interrupt

Output : FALSE if CDC already activated

TRUE otherwise

### 6.18 ILLEGALMSG

Format : ILLEGALMSG

Function: Process undefined message for a module.

An asynchronous message giving the MCB of the unde-

fined message is initiated at the CRT.

The undefined message is taken out of common vector

MSG.

### 6.19 ENTERINT

Format : ENTERINT BYTE (B1, B2)

Input : B1 - interrupt level

B2 - index of priority task

Output : FALSE if same interrupt already occupied

TRUE otherwise

Function: The interrupt on level Bl is enabled.

Upon occurrence of the interrupt the call of priority

task with index B2 is scheduled.

### 6.20 REMINT

Format : REMINT (B1)

Input : H1 - interrupt level

function: The enabled interrupt on level 81 is disabled.

### System Monitor

7

The system monitor is implemented as a system call and may be called when an internal error condition occurs, program limits are exceeded etc.

The system monitor generates an 'asynchronous output' message to a CRI module to indicate nature and location of the special condition.

Format : CALL SYSMON (A1, 81, 82);

Al any address value

B1 and B2 are any byte values

The Generated asynchronous CRT output has the following format:

\*\*\* SYSTEM MONITOR : XXXX YY ZZ

where XXXX = (A1) YY = (B1) ZZ = (B2)

It is recommended to set A1 to the stack pointer when calling the system monitor. In this case the displayed value XXXX will point to the memory location from where the system monitor was called.

B1 and B2 can be used to further specify the condition.

Up to now the system monitor is called with 81 representing the module number and 82 a specification within the module.

- B1 B2 condition
- 0 1 internal message buffer overflow
- 0 2 priority list overflow
- 0 3 periodic list overflow
- 0 4 use of illegal periodic index
- 0 5 use of illegal priority index
- 0 6 external message buffer overflow
- 0 7 double occupation of interrupt level
- 5 0 line printer output buffer overflow

Real-time Clock and Count Down Clock

The operating sytem provides two system clocks:

Real-time Clock (RTC):

The RTC is a public 4 byte vector. The least significant byte is RTC(3).

The value of the least significant bit is 1 millisecond.

The value of the least significant bit is 1 millisecond. Maximum time value is 4294967 sec = 71582 min = 1193 hrs = approx. 50 days.

The RIC is located in common memory and updated by SBC 1.

### Count Down Clock (CDC):

The operating system provides a CDC for each SBC.
The CDC can be activated by any module with the system call SETCDC.
The CDC has a length of two bytes and the least signifi-

cant bit is 1.86 micro sec.

System Loader and System Start

9

The system is started automatically after the loading is completed.

To load and start the system the following steps have to be performed:

- RESET and load ISIS-II
- enter monitor and change locations F1F4 and F1F5 to OABH and OCDH respectively
- RESET and load ISIS-II
- load and execute the loader by typing 'LOADER'
- the loader issues informative messages as it proceeds in the loading process.

The loader expects the code to be loaded into the SBCs to be in the files LUAD1,LUAD2 and LUAD3 respectively. These files are read from disk drive 0.

In case a file cannot be found, a warning message is typed and and the loading process continued.

The loader terminates if file LOAD1 cannot be found because the system cannot be started without SBC1 in this configuration. (In a general application, however, any number of SBCs can be loaded and started in any sequence.)

It should be noted that in a practical application of the operating system the on-board code would reside in ROM. 10 Example of a User Module

This section describes an example of a typical user module.

The example is given in form of a source listing in PL/M-80 to be compiled under ISIS-II.

DEMO:

00;

\$INCLUDE(:F1:SYDATE.SRC)
/\* include external definitions of system data \*/

\*\* DEMO SUBSTITUTIONS

A/ DCL

DM LIT '03', /\* module number of DEMO \*/

••

ENDSUBST LIT '0';

\*\* DEMO V

DEMO VARIABLE DATA

DCL

VARDATABEG BYTE,
(DMPERX, DMPRIORX) BYTE,

/\* storage for DM periodic and priority
indices\*/

(DMPERFL, DMPRIORFL) BYTE,

/\* flags controlling execution of DMPER
and DMPRIOR \*/

(DMPERADR, DMPRIORADR) ADDRESS,

/\* storage for entry addresses of periodic
and priority procedures \*/

VARDATAEND BYTE;

```
**
           DEMO CONSTANT DATA
**
*/
           DMTIMEINT (4) BYTE INITIAL (0,0,3,0E8H),
DCL
               /* time interval for DM periodic : 4 sec */
           DMMSG (4) BYTE INITIAL (4,DM, 10,7),
               /* MCB for message to module 4 */
           DMENDCONST BYTE;
$INCLUDE(:F1:SCEXT.SRC)
   /* include external definitions of system calls */
            *********
**
           DEMO UTILITY ROUTINES
**
*/
DMUT1:
           PROCEDURE;
           . .
           END DMUT1;
           PROCEDURE;
DMUTn:
           END DMUTn;
           DEMO PRIORITY PROCEDURE(S)
*/
DMPRIOR:
           PROCEDURE;
           IF DMPRIORFL = 0 THEN
               /* first call, find entry address of DMPRIOR */
             DMPRIORADR = PROCADR;
             DMPRIORFL = 1;
             RETURN;
             END;
           END DMPRIOR;
```

```
**********
* *
           DEMO PERIODIC PROCEDURE(S)
**
*/
DMPER:
           PROCEDURE;
           IF DMPERFL = 0 THEN
               /* first call, find entry address of DMPER */
             DO:
             DMPERADR = PROCADR:
             DMPERFL = 1;
             RETURN;
             END;
           END DMPER;
                 *********************
           DEMO START PROCEDURE
* *
*/
DMSTART:
           PROCEDURE;
           CALL CLEARDATA (. VARDATABEG, . VARDATAEND);
               /* clear variable data */
           CALL DMPRIOR;
           CALL DMPER;
               /* determine entry addresses of procedures */
           DMPRIORX = ENTERPRIOR(DMPRIORADR);
               /* enter priority call of DMPRIOR */
           DMPERX = PERACT(DMPERADR..DMTIMEINT(0));
               /* activate periodic call of DMPER */
           CALL UPDSTAT(DM, 3);
               /* set module's status
                   3 - existent and activated */
            IF NOT SEND(.DMMSG(0)) THEN RETURN;
               /* send demo message to module 4 with data
                  bytes 0,1,2 */
           DO B1 = 0 TO 2;
             MSGDATA(B1) = B1;
             END;
            . .
           END DMSTART;
```

NO OBJECT MODULE REQUESTED COMPILER INVOKED BY: PLM80 :F1:EXEC. SRC NOOBJECT PRGEWIDTH(80) PRGELENGTH(35) ISIS-II PL/M-80 V3. 0 COMPILATION OF MODULE EXECUTIVE

MODULES IN 1 CPTR \*/ MODULES IN SYSTEM \*/ PRIORITY CALLS \*/ COMPUTERS \*/ BUFFER \*/ /\* MAX NUMBER OF C
/\* MAX NUMBER OF M
/\* MAX NUMBER OF M
/\* LENGTH OF MSG B
/\* MAX NUMBER OF P SYSTEM DATA DECLARATIONS DECLARE LIT LITERALLY 'LITERALLY', DCL LIT 'DECLARE'S # INCLUDE(:F1:SYDATP. SRC) MAXSYSMOD LIT '24', MSGBUFLEN LIT '500', UPDATE: 1 MAXPRIOR LIT '8', MAXCPTR LIT '3', TRUE LIT 'OFFH', MAXMOD LIT '8', FALSE LIT '8', SMN LIT '1', RMN LIT '8', MN LIT '2', M LIT '3', EXECUTIVE: SCL N M 4

	INTVECTOR LIT '3000H', /* INTERRUPT VECTOR */ CPTR LIT '1', /* NUMBER OF HOST COMPUTER */	FIRSTMN LIT '8', /*	LASTMN LIT 'FIRSTMN+7', /* MN OF	EX LIT 'FIRSTMN', /*	LP LIT 'FIRSTMN+5', /* MN OF	CS LIT 'FIRSTMN+6', /* MN OF			*	***	*				SYSTEM WORK VARIABLES	BU(UPPER) AND BL(LOWER) OVERLAY ADDRESS VARIABLE A4	<b>/</b> *	DG DG	(MSG BASED ADRNSG) (4) BYTE,	(MSGDATA BASED ADRMSGDATA) (100) BYTE;	*	WORK AREAS FOR MSG CONTROL BLOCK (MSG) AND MSG		ADRMSG AND ADRMSGDATA ARE SET BY EXEC BEFORE THE			DCL RESTART BYTE PUBLIC;	**
11 11	11 11	11	11	11	11	11	11	11	II	11	11		H	11	11	11	11		11	11	11	11	11	11	11	11	11	11
												4						H									4	
												4						i)									9	

M

The state of the s

~

œ

```
SUSPEND MSG, SENT BY EXEC TO MODULE TO BE SUSPENDED **/
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          INTERNAL START MSG, SENT TO ALL MODULES IN CPTR
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                RESTART MSG, SENT TO MODULE TO BE ACTIVATED
                                                                                                                               MASK OF CURRENTLY ACTIVATED INTERRUPTS
FALSE IF START WITH SYSTEM RESET TRUE IF RESTART WITHOUT SYSTEM RESET
                                                                                                                                                                                                                                                                                                                                                                                                                                                                          DCL SYSMNØ(4) BYTE PUBLIC INITIAL (0, EX, 0, 4),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              SYSMN1(4) BYTE PUBLIC INITIAL (0, EX, 1, 4),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   SYSMN5(4) BYTE PUBLIC INITIAL (0, EX, 5, 4);
                                                                                                                                                                                                                                                                                                                                                                                              SYSTEM MESSAGES
                                                                                                                                                                                                                           MCB FOR DEBUG PURPOSES
                                                                                                                                                                                 DCL DEBUGMCB (4) L.'TE PUBLIC;
                                                                              DCL INTMASK BYTE PUBLIC;
                                                                                                                                                                                                                                                                                                                                                                                                                             ***
                                                                                                                                                                                                                                                                                                                                                                                                                                                                              a
```

12

<pre>\$EJECT \$ INCLUDE(&lt;:F1:SCEXT. SRC&gt;</pre>		********************	*****************		SYSTEM CALLS				PROCEDURE (P1, P2) EXTERNAL;	DECLARE (P1, P2) ADDRESS;	*	CLEAR MODULES VARIABLE DATA REGION	P1 - ADDRESS OF FIRST BYTE	P2 - ADDRESS OF LAST BYTE	/*	END CLEARDATA;	PROCEDURE BYTE EXTERNAL;	*	SEND MSG TO MODULE IN OTHER CPTR	*	END SENDEXT;	PROCEDURE EXTERNAL;	*	TRANSFER EXTERNAL MSG INTO OWN MSG BUFFER	*	END RECEXT;	PROCEDURE (P1) BYTE EXTERNAL;	DECLARE P1 ADDRESS;	*	SEND A MSG TO ANOTHER MODULE IN THE SYSTEM
\$EJECT \$ INCLUDE<	*	*******	*******	**		**	***	/*	CLEARDATA:								SENDEXT:					RECEXT:					SEND:			
	11	II	11	11	11	u	11	11	11		11	11	11	11				11	11				11	11					11	11
									+	N						Ø	=				O	-1				Ø	Ħ	N		
									30	31	:					32	33				46	33				36	37	23		

NAVAL POSTGRADUATE SCHOOL MONTEREY CALIF
A REAL-TIME OPERATING SYSTEM FOR SINGLE BOARD COMPUTER BASED DI--ETC(U) AD-A059 601 JUN 78 W NIEMANN NL UNCLASSIFIED 2 of 4 AD59 601

		11 1		P1 CONTRINS THE ADDRESS OF THE 1ST BYTE
		1		OF THE 115G CONTROL BLOCK.
		11		SEND RETURNS AN ADDRESS VALUE:
		11		- THE ADDRESS TO STORE THE 1ST DATA
		11		BYTE OF THE MSG
		11		- 0 IF THE RECEIVING MODULE DOES NOT
		11		EXIST OR IS NOT ACTIVATED.
		11		*
	N	11		END SEND;
		11		
	+	11	ACTIVATE:	PROCEDURE (P1.P2) BYTE EXTERNAL;
	O	11		DECLARE (P1, P2) BYTE;
		11		*
		11		ACTIVATE A CURRENTLY SUSPENDED MODULE.
		11		P1 - NUMBER OF MODULE TO BE ACTIVATED
		11		P2 - NUMBER OF CALLING MODULE
		u		ACTIVATE RETURNS A BYTE VALUE
		11		TRUE - OPERATION O. K.
		11		FALSE - CALLING MODULE NOT AUTHORIZED
		11		OR MODULE DOES NOT EXIST
		u		*/
•	OI	u		END ACTIVATE;
		H		
	4	11	SUSPEND:	PROCEDURE (P1, P2) BYTE EXTERNAL;
	N	11		DECLARE (P1, P2) BYTE;
		11		*
		H		SUSPEND A CURRENTLY ACTIVE MODULE.
		H		P1 - NUMBER OF MODULE TO BE SUSPENDED
		11		P2 - NUMBER OF CALLING NODULE
		11		SUSPEND RETURNS A BYTE VALUE
		11		TRUE - OPERATION O. K.
		11		FALSE - CALLING MODULE NOT AUTHORIZED

OR MODULE CRANOT BE SUSPENDED	/*	END SUSPEND;		PROCEDURE (P1) BYTE EXTERNAL;	DECLARE P1 BYTE;	**	CHECK WHETHER A MODULE IS ACTIVATED	P1 - MODULE NUMBER	ACTIVE RETURNS A BYTE VALUE	TRUE - MODULE IS ACTIVE	FALSE - MODULE IS NOT ACTIVE	/*	END ACTIVE:		PROCEDURE (P1, P2) BYTE EXTERNAL)	DECLARE (P1, P2) ADDRESS;	*/	ACTIVATE PERIODIC CALL OF A PROCEDURE	P1 - ADDRESS OF PERIODIC PROCEDURE	P2 - ADDRESS OF TIME INTERVAL	PERRCT RETURNS THE INDEX OF THE PERIODIC	/*	END PERRCT;		PROCEDURE (P1, P2) EXTERNAL;	DECLARE P1 BYTE, P2 ADDRESS;	*/	CHANGE THE TIME INTERVAL OF A PERIODIC	P1 - PERIODIC INDEX	P2 - ADDRESS OF NEW TIME INTERVAL
				ACTIVE:											PERACT:										PERCHG:					
II	11	11	11	11	11	11	u	11	11	11	11	11	11	11	11	11	11	II	11	11	11	11	11	u	11	11	ıı	11	11	11
		Ø		4	N								N		4	N							N		4	0				
		45		46	47								48		49	50							ដ		52	53				

94c

END PERCHG			DECLARE P1 BYTE;	**	SUSPEND A PERIODIC	P1 - PERIODIC INDEX	*	END PERSUSP;		ENTERPRIOR: PROCEDURE (P1) BYTE EXTERNAL;	DECLARE P1 ADDRESS;	*	ENTER NEW PRIORITY CALL IN PRIORITY LIST.	P1 - RODRESS OF PRIORITY PROCEDURE	ENTERPRIOR RETURNS THE PRIORITY INDEX	/*	END ENTERPRIOR:		PRIORITY: PROCEDURE (P1) EXTERNAL:	DECLARE P1 BYTE;	*	REQUEST CALL OF A PRIORITY PROCEDURE	P1 - PRIORITY INDEX	/*	END PRIORITY:		PRIORITYINT: PROCEDURE (P1) EXTERNAL;	DCL P1 BYTE;	END PRIORITYINT;		REMPRIOR: PROCEDURE (P1) EXTERNAL)
11 11	-	1	11	u	11	11	11	11	11	II	11	11	11	11	11	u	11	11	11	11	11	11	u	11	11	11	11	u	11	11	u
N	*	1	N					N		4	N						N		ч	N					N		7	N	N		त
3	60	3	36					25		28	29						68		61	62					63		64	9	99		67

DECLARE P1 BYTE;  /*  REMOVE PRIORITY CALL FROM PRIORITY LIST  P1 - PRIORITY INDEX  */  END REMPRIOR;	SETBIT: PROCEDURE (P1,P2) EXTERNAL; DECLARE P1 BYTE,P2 ADDRESS; /* SET BIT P1 IN BYTE P2 */ END SETBIT;	CLEARBIT: PROCEDURE (P1,P2) EXTERNAL; DECLARE P1 BYTE,P2 ADDRESS; /* CLEAR BIT P1 IN BYTE P2 */ END CLEARBIT;	T: PROCEDURE (P1,P2) BYTE EXTERNAL; DECLARE (P1,P2) BYTE;  /* CHECK BIT P1 IN BYTE P2  RETURN TRUE IF BIT IS SET, FALSE OTHERWISE  */	END BIT; SYSMON: PROCEDURE (P1, P2, P3) EXTERNAL; DECLARE P1 ADDRESS, (P2, P3) BYTE;
		٦ ٦		· · · · · ·
α α	40 0	40 0	40	0 40
8 6 8	78 71 72 72	5 5	3.2	25 86 88

	SYSTEM MONITOR	MAY BE CALLED WHEN AN INTERNAL ERROR	CONDITION EXISTS.	P1 - LOCATION OF ERROR	P2, P3 - SPECIFICATION OF ERROR		END SYSMON:	TO A POUTE CATERAIGH	PROCEDURE (FI) BYIE EXIERMAL)	4DDRESS;		CHECK RTC AGAINST INPUT TIME	S OF TIME IN RTC FORMAT	RETURN TRUE IF INPUT TIME < RTC	FALSE OTHERWISE			PROCEDURE (P1) BYTE EXTERNAL;	YTE;		RETURN A BYTE WITH A '1' SHIFTED INTO	- P1				PROCEDURE (P1) EXTERNAL;	DCL P1 ADDRESS;		PACK MCB INTO MSGBUFFER		MCB;	
						*			INECHA: FRUC	DECLARE P1 ADDRESS;	*/	CHECK RT	P1 - ADDI	RETURN TI	II.	**	END TIMECHK	SHFT: PR(	DECLARE P1 BYTE	*	RETURN A	BIT POSITION P1	\*	END SHFT;		PACKMCB: PROC	DCL I	*/	PRCK	/*	END PACKNCB	
1	11	11	11	11	11	11	11	1	18	11	11	u	11	11	11	11	u	11	11	11	11	11	11	11	u	u	H	11	11	11	11	11
							0	٠,	1	N							N	7	N					N		7	N				Ø	
							ă	3 8	No.	83							8	8	86					87		88	83				38	

- 33 -

94**f** 

SETPERTIME: PROCEDURE (P1) EXTERNAL;	DCL P1 BYTE:	*	SET NEXT CALL TIME OF A PERIODIC	/*	END SETPERTIME;	UPDSTAT: PROCEDURE (P1, P2) EXTERNAL;	DECLARE (P1. P2) BYTE;	*	UPDATE MODULE STATUS	P1 - MODULE NUMBER	P2 - STATUS	/*	END UPDSTAT;	PROCADR: PROCEDURE ADDRESS EXTERNAL:	*/	RETURN ADDRESS OF BEGIN OF CALLING	PROCEDURE	/*	END PROCADRO	CONVASC: PROCEDURE (P1) EXTERNAL)	DECLARE P1 BYTE;	*/	CONVERT 2 DIGIT HEX NUMBER INTO	2 ASCII CODES	P1 - NUMBER	CUITPUT IN SYSTEM VARIABLE BU, BL	/*	END CONVASC:	ILLEGALMSG: PROCEDURE EXTERNAL;	*	PROCESS ILLEGAL MSG RECEIVED BY
				11	11					11	11	11			u	11	11	11	11		11	11	11	11	11		11		11	11	11
_	2				~	7				ï			~	7					~	H	" ~						"		7		"
"					``	``							``	.,					` '	,,									**		
91	92				88	7	38						8	97					88	66	166							161	162		
												-	3	4	_										91	+g					

A MODULE	\*	END ILLEGALMSG;	DEBLK: PROCEDURE EXTERNAL;	*	NOVE POINTER B1 TO 1ST NON-BLANK IN NSGDATA	STARTING AT LOCATION MSGDATA(B1)	*	END DEBLK;	GETNUM: PROCEDURE BYTE EXTERNAL;	**	CONVERT ASCII CODES IN MSGDATA TO HEX NUMBER	START AT MSGDATA(B1)	CONVERTED NUMBER IS LEFT IN A4	RETURN TRUE IF NUMBER IS O. K. , FALSE OTHERWISE	*	END GETNUM:	SETCDC: PROCEDURE (P1, P2) BYTE EXTERNAL;	DCL (P1, P2) ADDRESS;	END SETCDC;		ENTERINT: PROCEDURE (P1, P2) BYTE EXTERNAL;	DCL (P1, P2) BYTE:	END ENTERINTS		REMINT: PROCEDURE (P1) EXTERNAL)	DCL P1 BYTE;	END REMINT;	END;	
11	11	11	11	11	11	ıı	11	11	11	11	11	II	tı	11	11	11	11	u	11	11	II	11	u	11	II	11	11		
		N	त					N	त							N	4	N	N		त	Ø	0		त	N	N	4	
		163	164					105	106							187	168	109	110		111	112	113		114	115	116	117	

### APPENDIX C

USER'S MANUAL FOR DEBUG FUNCTIONS

### TABLE OF CONTENTS

1	General	5
1.1	Introduction	5
1.2	Abbreviations and Conventions	2
2	Initiation of Debug Mode	3
3	Use of Debug Functions	3
3.1	Inspect and Change	4
3.2	Memory Dump	5
3.3	Snapshot	5
3.4	Message Simulation	8
3.5	Message Extraction	9
3.6	Periodic Inspect/Dump	9
3.7	Hardcopy	10
4	Termination of a Debug Function	11
5	Termination of Debug Mode	11
6	Summary of Debug Commands	12

### 1 General

### 1.1 Introduction

The implemented debug functions are specifically designed to ease the debugging of user modules running under the control of the real-time operating system without interference with the operation of the system.

Since the debug module (DB) is a user module itself, the functions are available only if this module is integrated into the system.

The debug module communicates with the user using the CRT, therefore the integration of the CRT module (CS) is also necessary.

If the user wishes to obtain hard copies of the debugging results, the line printer module (LP) has to be integrated, too.

Inputs at the CRI are terminated with the RETURN key.
The line editing functions of the CS module can be used.

### 1.2 Abbreviations and Conventions

All inputs and outputs of the debug functions are in the hexadecimal number system.

Numbers in this text are decimal, hexadecimal numbers are trailed by the letter 'H'.

DB - module identification of debug module

CS - module identification of CRT module

LP - module identification of line printer module

RMN - receiving module number

SMN - sending module number

MN - message number

ML - message length

MCB - message control block
MCB consists of 4 bytes:

RMN

SMN

MN

ML

SBC - single board computer

hard copy - printer output of debug function

Initiation of the Debug Mode

2

The debug mode is entered with the input of 'Control-D' at the CRI keyboard. The debug module, if integrated, responds with 'DEBUG CPTR:' and expects the input of the number of the SBC which the user wishes to debug.

Depending on the input the following system reactions are possible:

- SBC number not specified:
  A '?' is typed and 'DEBUG CPTR:' is repeated.
- Requested SBC is not activated:
   'CPTR NOT ACTIVATED, DEBUG TERMINATED' is typed and the debug mode is left.
- Requested SBC already debugged from another SBC:
  'CPIR ALREADY DEBUGGED, DEBUG TERMINATED' is typed and the debug mode is left.
- Input accepted: The system responds with a new line and the prompt character of the debug function '>'.

Now any of the debug commands described in Section 3 may be entered.

Use of the Debug Functions

In this section the inputs and outputs of all debug functions are described. Furthermore, some hints for the usage are given. In the following description, system outputs are given prefaced by: 'SYSTEM OUTPUTS'.

Debug commands are prompted with '>'.

If a debug command requires more than 1 input, these commands are prompted with ':'.

Illegal inputs are reported with the output of '?'.

### 3.1 Inspect and Change

This function allows the display of byte and address values and is invoked by 'A' (for address) or 'B' (for byte). The 'C' command allows the change of the address or byte value displayed last.

### Address inspection:

>A XXXXcr 'AXXXX : YYYY

Note: The contents of the address value is displayed in the format a user would read it. The internal representation in memory is in the reverse byte form.

### Byte inspection:

>B XXXXcr 'BXXXX : YY

## Change of contents:

>A XXXXCr 'AXXXX : YYYY
>'CZZZZcr 'AXXXX : ZZZZ
>'
>B XXXXCr 'BXXXX : YY
>'CZZcr 'BXXXX : ZZ

Note: A change command can be repeated, it always refers to the last inspected memory location (address or byte).

The change command is only legal following an inspect command.

The input of address values is in user readable format and not in internal machine representation.

Note: If the last command has been 'A', 'B' or 'C' then the input of RETURN only will display the next byte or address value.

### 3.2 Memory Dump

With this function the user can display a contiguous portion of memory specified by a begin and end address.

The dump allows the display of either byte or address values.

The range of the dump can be specified in 2 different ways:

DBXXXX,Z or DBXXXXX-YYYY

where XXXX is the begin address

YYYY is the end address

Z is the number of values to be dumped

Dump of byte values:

DBXXXX,Zcr
'BXXXX : VV VV VV VV ... (up to 16 byte values/row)'

Dump of address values:

DAXXXX,Zcr
'AXXXX : VVVV VVVV VVVV ... (up to 8 address values/row)'

### 3.3 Snapshot

This function allows the display ('snapshot') of

- CPU registers/flags
- up to 16/32 contiguous address/byte memory locations
- independently specified address or byte memory locations

before the execution of a specified instruction. Furthermore, the taking of the snapshot can be conditional in the sense that a specified memory location has to have a specified value at the time of the snapshot.

The taking of the snapshot and the display of the results do not influence the operation of the system.

The snapshot remains activated until terminated as described in Section 4. The function is executed each time the specified snapshot instruction is executed.

Note: Because of the snapshot method of inserting a trap instruction it is not possible to activate a snapshot in a ROM section of the program.

#### Input format:

S XXXX parameter list cr

where 'S' is the debug command and XXXX the address of the first byte of the snapshot instruction.

Note: If the snapshot address is not set to the first byte of an instruction, an erroneous program action may occur.

The parameter list can be any combination of the following inputs:

- R This parameter requests the display of all CPU registers and flags.
- D
  This parameter specifies a contiguous region in memory whose contents at the time of the snapshot is to be displayed. The format of the parameter is identical to the debug function 'dump' described in Section 3.2.

  The region to be dumped is limited to 10H/20H address/byte values.
- A or B
  These parameters have the form AXXXX or BXXXX
  and cause the display of address/byte value at
  location XXXX respectively. A maximum of 5 locations, either address or byte, can be specified.
- M This parameter has the format

#### MXXXX-YY

where XXXX is a byte location in memory and YY the specified contents. If this condition is set and the contents of XXXX is not YY, then the snapshot is not taken. This feature allows to check the state of the system at a specified iteration of a loop (M parameter is the index variable) or to monitor the occurence of a specified input to the system.

#### Output format:

The snapshot function has no outputs until the specified instruction is executed. If no parameter was specified in the 'S' command then only the standard output occurs:

'SNAPSHOT AT XXXX : YY YY YY'

where XXXX is the snapshot address and
YY YY YY is the snapshot instruction (max 3 bytes).

The output initiated by the parameters is self-explanatory.

If during the output processing of the snapshot results another snapshot occurs, it will be suppressed. The number of suppressed snapshots is typed.

#### 3.4 Message Simulation

This function allows the generation of real-time messages and is useful to trigger processes for which the initiating module is not integrated or the initiating input is not available.

After the input of the debug command 'MS', the function prompts the user for inputs in order to construct the message to be simulated.

The input format is shown by the following example.

### Input format:

>MScr
'RMN:'17cr 'SMN:'8cr 'MN:'10cr 'ML:'6cr
'DATA BYTES:'10cr ':'20cr
'MSG SENT
>'

The above commands would cause the sending of a message with the receiving module number 17H, the sending module number 8, message number 10H, message length 6 with two data bytes 10H and 20H.

The minimum message length is 4 (length of MCB).

Maximum message length is 24H: MCB and 20H data bytes.

The input of data bytes is prompted automatically depending on the specified message length. The message is sent as soon as the input is completed.

After the prompt symbol reapears the same message can be repeated with the input of the RETURN key only.

In the case of an illegal input during the input of the message, the same (wrong) input is prompted again. If the specified message cannot be sent by the system because the receiving module is not activated, then 'MSG NOT SEND' is typed.

#### 3.5 Message Extraction

This function allows the interception of any specified real-time message at any time.

The user can specify any combination of RMN, SMN, MN and ML. If the specified message is to be processed by the receiving module, the message (including data bytes) is typed on the CRT.

The input/output formats are illustrated with the following example.

### Input format:

>MXCr

'RMN: '17cr 'SMN: '10cr 'MN: 'cr 'ML: 'cr

An input request answered with the input of RETURN only has the meaning of 'not specified'. In the above example all messages from module 10H to module 17H are extracted because MN and ML are not specified and can take on every value.

### Output format:

'RMN: 17 SMN: 10 MN: 11 ML: 16 DATA: VV VV .....

VV VV .....

(wo to 16 bytes/line)

### 3.6 Periodic Inspect/Dump

This feature repeats the output of an 'inspect' or 'dump' function at maximum system speed and allows therefore a continuous 'look' into memory in order to observe changes of variables, status bits etc.

These functions are initiated by the same commands described in Section 3.1 and 3.2 with the difference that they are preceded by the letter 'P' (for periodic).

#### Examples:

PDBXXXX, ZZcr

PDAXXXXCE

Note: The number of values to be dumped by the periodic dump is restricted to 10H/20H address/byte values.

A larger range is cut down to this maximum length by the debug function.

'Maximum system speed' is ususally determined by the speed of the output media.

### 3.7 Hard Copy

This function allows the user to obtain hard copies of the debugging results.

If a hard copy is requested, the outputs of

- memory inspection
- memory dump
- snapshot
- message extraction
- periodic functions

are directed to the line printer.

The user may switch at any time between the CRT and line printer output.

Default output device is the CRI.

Input format:

>Hcr

The first 'H' command switches output to the line printer, the next 'H' command switches back to CRT and so forth.

If output is switched to the line printer, the print head is positioned at the top of a new page.

In case the line printer is not connected properly, an asynchronous message 'LINE PRINTER NOT CONNECTED' is typed on the CRT and the 'H' command is disregarded.

Termination of a Debug Function

A selected debug function terminates in 3 different ways:

- a) The function terminates itself after the inspection of a memory location. In this case the system shows the prompt character '>'.
- b) If a debug function requires more than 1 input or is executed periodically, it can be terminated with the input of 'Control-I' at any time. The system responds with the prompt character.
- c) At any time within the debug mode the CRT interruption 'Control-I' can be used to disconnect the debug module from the CRT. In this case, however, not only the currently activated debug function is stopped, the debug mode is terminated as well. The only system reaction is the positioning of the cursor to the beginning of a new line.

#### Termination of Debug Mode

The debug mode can be terminated in 2 different ways:

- a) Use of the debug command 'T'. This input can be made when the prompt character is displayed and a regular command is expected. The system responds with 'DEBUG TERMINATED' and leaves the debug mode.
- b) At any time the procedure decribed in 4 c) can be used.

Summary of Debug Commands

### Debug Functions:

A - inspect address value

8 - inspect byte value

C - change address or byte value

D - memory dump

H - hardcopy

MS - message simulation

Mx - message extraction

P - inspect/dump in periodic mode

S - snapshot

T - terminate debug mode

### Other Debug Inputs:

Control-D - initiate debug mode

Control-I - terminate debug function

Control-I - CRI interruption (terminates debug mode)

# APPENDIX D

PROGRAM DESCRIPTION OF DEBUG MODULE

# TABLE OF CONTENTS

1	General	3
1.1	Introduction	3
1.2	Abbreviations and Conventions	3
1.3	Facilities of the DB module	4
1.4	Module Program Organization	4
2	Input Real-time Messages	6
2.1	Start	6
2.2	Input Text from CRT	6
2.3	Terminate I/0	6
2.4	Start Debug	6
2.5	External Start Debug	7
2.6	Terminate Debug Function	7
2.7	Output Acknowledge	7
8.5	Message Extraction	7
2.9	Line Printer not Connected	8
3	Processing	9
3.1	Inspect and Change	9
3.2	Memory Dump	10
3.4	Snapshot	10
3.4	Message Simulation/Extraction	13
3.5	Hardcopy	14
3.6	Periodic Inspect/Dump	15
3.7	Real-time Messages	15

3.7.1	Start	15
3.7.2	Input Text from CRT	15
3.7.3	Terminate I/0	16
3.7.4	Start Debug	17
3.7.5	External Start Debug	17
3.7.6	Terminate Debug Function	17
3.7.7	Output Acknowledge	17
3.7.8	Message Extraction	18
3.7.9	Line Printer not Connected	18
4	Output Real-time Messages	19
4.1	Synchronous Output	19
4.2	CRT Input Request	19
4.3	Terminate CRT Input	19
4.4	Activate Debug	19
4.5	New Line	20
4.6	Beginning of Line	20
4.7	Synchronous Output with Output Acknowledge	20
4.8	External Start Debug	20
4.9	Start Message Extraction	21
4.10	Stop Message Extraction	21
4.11	Initialize Line Printer	21

#### 1 General

#### 1.1 Introduction

The debug module (module identification: DB) allows the debugging of all modules in all single board computers under real-time conditions.

This means that the use of a debug function does not influence the operation of the system.

Since each single board computer has its own kernel of the debug module, it is possible that several users debug different program parts at the same time.

There are no special requirements for the integration of DB except that at least one CRT module has to be integrated, too.

The CRT and the line printer are the I/0 media of the debug module. Real-time messages are used to communicate with the respective modules.

#### 1.2 Abbreviations and Conventions

All numbers in this segment of text are decimal except as indicated otherwise.

SBC - single board computer

MCB - message control block

RMN - receiving module number

SMN - sending module number

MN - message number

ML - message length

CS - CRT module, module number in SBC 1/2/3 : 06/14/22

EX - executive, module number in SBC 1/2/3: 00/08/16

DB - debug module, module number in SBC 1/2/3: 07/15/23

LP - line printer module, module number in SBC 1/2/3: 05/13/21

### 1.3 Facilities of the Debug Mocule

The debug module offers the following debug functions under real-time conditions:

- inspection and change of byte and address values
- memory dump of byte and address values
- snapshot with
  - . register dump
  - . dump of specified single memory locations
  - . dump of one contiguous region of memory
  - . specified condition
- message simulation
- message extraction
- periodic inspection/aump
- choice between CRT and line printer output

#### 1.4 Module Organization

Because of its size the DB module has been split up into 9 parts in order to ease program development and program maintenance under the ISIS-II operating system. The single parts of DB are program modules in the sense of PL/M-80 and ISIS-II.

Each of the 9 parts has a seperate source file while the object code is kept in one object library which represents the object code of the DB module.

In the following the parts of DB are described briefly:

- DBMOD1: This part is the entry for real-time messages in DB.

  It also includes the PUBLIC definitions of all DB data and the START procedure.
- DBMUD2: This part processes all input messages from the CRT module. It
  - performs the initialization of the debug mode
  - relays debug requests to responsible debug modules in other SBC's if necessary
  - distributes debug inputs and messages to the respective debug function.

The program and data organization of DBMOD2 allows easy to implement additions to the debug functions.

DBMOD3: Memory inspection and change

DBMOD4: Memory dump

DBMOD5: Snapshot

DBMOD6: Message simulation and extraction

DBMOD7: System test

DBMOD8: DB utility routines

DBM009: DB utility routines

### 2 Input Real-time Messages

DB obtains all inputs from the CRT module.

The format of the debug inputs is described in the user's manual for the debug functions.

In this section the format and contents of all input messages is described. The respective process can be found in Section 3.7.

#### 2.1 Start

RMN : DB SMN : EX MN : 00 ML : 04

This message informs DB that the system has been started.

### 2.2 Input Text from CRT

RMN : DB SMN : CS MN : 20

ML : depends on length of input text

DATAO-

DATAn : ASCII input characters

This message transmits input text from the CRT to DB.

### 2.3 Terminate I/O

RMN : DB SMN : CS MN : 22 ML : 04

This message is received when the CRT interruption input was made during activated debug mode.

### 2.4 Start Debug

RMN : DB SMN : CS MN : 23 ML : 04

This message is received when a 'Control-D' input was legally made at the CRI keyboard: a user wishes to activate the debug mode.

#### 2.5 External Start

RMN : DB

SMN : DB module in other SBC

MN : 24 ML : 05

DATAO: number of CS module for debug inputs and outputs

This message is received when the host SBC is to be debugged from another SBC.

DATAO contains the module number of a CS module to communicate with.

### 2.6 Terminate Debug Function

RMN : DB SMN : CS MN : 25 ML : 04

This message informs DB that a 'Control-I' input was made at the CRI keyboard.

With this input the user wishes to terminate the currently selected debug function.

### 2.7 Output Acknowledge

RMN : DB

SMN : CS or LP

MN : 26 ML : 05

DATAO: tellback code

This message is sent by CS or LP in order to indicate that the output requested by DB has been completed. The returned acknowledge code is the code sent to CS or LP with the 'output with acknowledge' message.

#### 2.8 Message extraction

RMN : DB SMN : EX MN : 27

ML : depends on length of extracted message

DATA -

DATAn: MCB and data bytes of extracted message

This message is received when message extraction is activated and the executive has intercepted a message specified for extraction.

# 2.9 Line Printer not Connected

RMN : DB SMN : LP MN : 28 ML : 04

This message informs DB that the line printer is not connected to the system.

#### 3 Processing

This section describes the function of the debug facilities and the process of the received real-time messages.

#### 3.1 Inspect and Change

The process of this function is contained in DBMOD3 and consists of 3 procedures:

DBINSPECT, DBCHANGE and TYPEINSP.

#### DBINSPECT is called when

- a) a A,B,PA or PB command has been entered at the CRT
- b) RETURN only was entered at the keyboard and the last activated function was A,B or C
- c) an acknowledge message was received and the last command was PA or PB.

In case a) the address of the memory location is decoded from the input message. If the input was not legal, the error indication '?' is sent to the CRI module. If a periodic output was requested, the cursor is positioned at the beginning of the input line and TYPEINSP is called with acknowledge request. Otherwise TYPEINSP is called with the output followed by the prompt character.

DBCHANGE is called for the execution of the 'C' command. If the previous input has not been A, B or C, then the change is not legal and an error is indicated. Otherwise the new contents is stored and TYPEINSP is called to type address and new contents.
DBCHANGE terminates with the issue of the prompt character.

TYPEINSP performs the output of the current 'inspect and change' address and the respective contents (byte or address).

Depending on the state of the flag DBIELLBACK this is done with or without acknowledge request.

### 3.2 Memory Dump

The memory dump is performed by DBMOD4 and consists of 5 procedures:

DRDUMP, DUMPDECODE, INITDUMP, DUMPLINE and DUMP.

DBDUMP is the entry procedure. From here the respective routines to set up, execute or repeat the dump (in case of periodic dump) are called.

DUMPDECODE determines the mode, byte or address, and the range of the dump.

In case of an illegal input, a logical 'FALSE' is returned, otherwise a 'TRUE' is returned.

DUMPDECODE is called from DBDUMP and DBSNAP (see 3.3).

DUMP is the procedure which actually performs the dump from DUMPBEG to DUMPEND.

DUMP is called from DBDUMP and SNAPSHOT.

The procedure keeps calling DUMPLINE until the flag DUMPDONE becomes 'TRUE'.

INITDUMP sets up the format of the output line. It packs the address of the first value typed on a line and determines the number of values typed on a line depending on the mode of the dump.

DUMPLINE packs the contents of the memory locations starting at the current address. Up to ENDLINE values are packed for a line. DUMPLINE uses the utility procedures PACKBYTE and PACKADR to convert the contents of memory locations into ASCII codes and pack into DB output buffer.

#### 3.3 Snapshot

The snapshot function constitutes DBMOD5 and consists 4 procedures.

Basically the snapshot is enabled by replacing the operation code of the instruction at the snapshot address by the trap instruction RSI4. The actual snapshot is taken when this instruction is executed and an 'interrupt' occurs.

The display of the snapshot data is performed with a priority call scheduled during the process after the execution of the trap instruction.

The snapshot remains activated until it is terminated by a command. At this time the original instruction is restored at the snapshot address.

If the system is stopped with an activated snapshot and started again without new loading, the snapshot address still contains the RSI4 instruction.

The start procedure of DB checks for an activated snapshot before system stop and restores the original instruction if necessary.

DBSNAP initiates the snapshot and performs the following functions:

- reset all relevant snapshot data
- process CRI input
  - . 'R' set REGFL
  - . 'M' set SNCONDADR to condition address and SNCONDSAVE to the condition set CONDFL
  - 'D' determine address and type of dump (procedure DUMPDECUDE is used) set DUMPFL
  - . 'A'/'B' determine address and type of the requested variables and store in SNVAR update SNVARX (index to SNVAR)

In case that an illegal input is encountered during the process of the CRT input, the display of '?' and the prompt character is initiated and program control is returned.

- determination of the length of the snapshot instruction
- insertion of the instruction in the execution table SNEXTBL (see structure of SNEXTBL below)
- insertion of the address of the next instruction into SNEXTBL (= snapshot address + instruction length)
- insertion of RST4 at the snapshot address

Organization and function of the snapshot execution table:

	*****		
0	* E1 *	POP	H
	******		
1	* F1 *	POP	PSW
	******		
5	* C1 *	POP	В
	*****		
3	* D1 *	POP	D
	******		
4	* E1 *	POP	H
	*****		
5	* 33 *	INX	SP
	*****		
6	* 33 *	INX	SP
	*****		
7	* FB *	EI	
	*****		
8	* 00 *		replaced
	*****		
9	* 00 *		snapshot
	*****		
10	* 00 *		instruction
	****		
11	* C3 *	JMP	
	******		
12	*		address of instruction
	*****		
13	*		after snapshot instruction
	******		

At the end of the snapshot 'interrupt' process, program control is transferred to the first byte of the execution table where

- the stack is updated
- interrupts are enabled
- the replaced instruction is executed
- program control is transferred back to the 'interrupted' program.

SNAPINT is the procedure which services the snapshot interrupt. The return from this procedure takes place as described above.

If the specified condition is not, met control is returned.

If there is already a snapshot in progress, i.e. SNAPFL is 'TRUE', a counter is incremented in order to count multiple occurrences of the same snapshot.

If none of the above applies, SNAPFL is set and the current value of the requested parameters are saved if the respective flag (SNVARX, DUMPFL or REGFL) is set.

After saving of the specified values and scheduling a priority call for procedure SNAPSHOT, SNEXTBL is executed.

The procedure SNAPSHOT outputs the snapshot data to the CRT. This is done with 'synchronous output with acknowledge' message to the CS module.

The acknowledge mode prevents a possible overflow of the CRT output buffer.

The returned acknowledge code is used to distribute the process sequentially through SNAPSHOT.

Since this process is straightforward, it is not described in detail.

At the end of SNAPSHOT the flag SNAPFL is reset to allow the output of the next snapshot.

Procedure SNAPTERM has the task of terminating the snapshot function. It is called when the 'terminate debug function' message is received or at system start when the system was stopped before with an activated snapshot.

SNAPTERM restores the original instruction at the snapshot address.

### 3.4 Message Simulation/Extraction

DBMOD6 consists of the functions message simulation and message extraction. These functions are handled in the procedures DBMSX, DBMSXINP and DBEXTRACT.

DBMSX is the entry procedure for DBMOD6 and is called from the message entry of the DB module. If flag DBTELLBACK is 'TRUE' the procedure DBEXTRACT is called, otherwise DBMSXINP is called.

DBMSXINP processes the CRT input message for simulation and extraction  $\cdot$ 

In order to obtain all necessary inputs, the procedure initiates a dialog with the operator at the CRI.

All input/output is performed with real-time messages to and from the CS module.

In both cases, simulation and extraction, the dialog yields a message control block.

Illegal inputs are rejected and prompted again.

The MCB is stored in DEBUGMCB, a table located in the system data area to allow access by the executive in order to check for messages to be extracted.

After completion of the MCH input, the process splits.

#### Message simulation:

The dialog continues until the input of data bytes as specified in the MCB is completed. After this, the constructed message is sent.

If the message could not, be sent the output of 'MSG NOT SENT' is issued.

If the sending of the message was successful 'MSG SENT' is typed.

The function terminates with the sending of the prompt character.

If the last function has been 'message simulation' and the input of RETURN only is received, the sending of the same message is repeated.

#### Message extraction:

The extraction is initialized with the sending of an extraction message to the executive.

DBEXTRACT processes extraction messages from the executive. The CRT output is done using output with acknowledge messages. The process is straightforward and is not described here.

### 3.5 Hardcopy

The 'H' command is processed in the procedure DBHARDCOPY which is part of DBMOD8.

If flag HARDCOPY is 'IRUE' then it is set to 'FALSE'.

If flag HARDCOPY is 'FALSE' an 'initialize line printer' message is sent to the LP module and HARDCOPY is set to 'IRUE'.

Before leaving the procedure, a new debug command is prompted.

In case the line printer is not connected to the system, a 'line printer not connected' message will be received by DB. Upon receipt of this message the flag HARDCOPY is set to 'FALSE'.

HARDCOPY controls the output media in procedure DBSYNCPRH. Depending on the state of HARDCOPY, the receiving module for the 'synchronous output' message is LP or CS.

### 3.6 Periodic Inspect/Dump

The inspect and dump functions can also be executed as periodic functions, i.e. the output is repeated with the highest frequency the system allows.

The output of the functions is sent to LP or CS with acknowledge request. Upon receipt of the acknowledge message the same requested output is repeated with the current contents of the specified locations.

### 3.7 Real-time Messages

In this section the process of the real-time messages is described.

#### 3.7.1 Start

This message is processed in procedure DBSTART. The following operations are performed:

- if a snapshot has been activated before the last system stop (DBFUNCTION = 4), then call SNAPTERM (restore snapshot instruction)
- clear DB variable data
- clear snaoshot data
- set module status (existing and activated)
- determine start address of procedures DBINPUT and SNAPSHOT
- enter priority call for SNAPSHOT.

The DB module is then ready to be used.

#### 3.7.2 Input Text from CRT

This message is processed by the procedure DBINPUT which constitutes DBMOD2.

The execution of DBINPUT is controlled by a case statement and the variable CRIINPUT.

### CRTINPUT = 0:

Input from CRT must be the initiation of the debug mode. 'CPTR:' is typed on the CRT, CRTINPUT is set to 1 and a 'CRT input request' message is sent to CS.

#### CRTINPUT = 1:

The input of the number of SBC which the user wishes to debug is expected now.

If the input is not legal, then '?-CPTR:' is typed and the input request repeated.

If the number is 0 or its own SBC number, the procedure DBREQ is called to initiate the debug mode. In DBREQ the control variable is set to 2 and a prompt for a debug command is issued. Otherwise the number of the debug module in the target SBC is computed. If the respective module is not activated, 'DEBUG MODULE NOT ACTIVATED' is typed. If the module is active, a 'start debug external' message is sent to the external debug module. The module number of the own CS module is transmitted in

The control variable is reset to 0.

#### CRTINPUT = 2:

The input is the selection of a debug function.

DBINPUT decodes the function identifier and calls the respective procedure to process the input.

The variable DBFUNCTION contains an index which determines the selected debug function.

CRIINPUI is set to 3.

the first data byte.

#### CRTINPUT = 3:

This input is caused by the selected debug function itself. Program control is routed to the process of the currently activated function.

#### 3.7.3 Terminate I/O

This message informs the DB module that the connection to the CRI has to be terminated (input of 'Control-I' at the CRI).

The input control variable is reset to 0 in order to accept a new initialization of the debug mode. Any pending debug function and the debug mode are terminated.

### 3.7.4 Start Debug

This message informs DB that a 'Control-D' input has been made at the CRT.

This input is processed in procedure DBINPUT with control variable CRTINPUT = 0 (see Section 3.7.2).

### 3.7.5 External Start Debug

This message is sent by a debug module in an other SBC and informs DB that there is an external debug request from an other SBC.

The message is processed in procedure DBEXIREQ.

After saving the module number of the connected CRT in the other SBC in DBCRT procedure, DBREQ is called to further process the debug request.

#### 5.7.6 Terminate Debug Function

This message is received after a 'Control-I' input at the CRI and it serves the purpose of terminating the currently activated debug function.

The process takes place in procedure DBTERMPER.

If no debug function is activated (i.e. the prompt character is displayed), an error message is sent to CS.

If the activated function is 'message extraction', a message is sent to EX to stop the extraction.

An activated snapshot is terminated with a call of SNAPTERM.

After resetting of all relevant flags, the prompt character is displayed.

#### 3.7.7 Output Acknowledge

This message is received when an output with acknowledge initiated by DB is completed.

The returned acknowledge code is saved in DBTBCODE and the flag DBTELLBACK is set to 'IRUE'. Then DBINPUT is called which passes program control to the activated function controlled by CRIINPUT and DBFUNCTION.

#### 3.7.8 Extraction Message

This message transmits an extracted message from EX to the 'message extraction' function in DB.

If an extraction output is currently active (DBEXTRACT = 'TRUE') a counter is incremented and DBEXTRACT is not called. Otherwise DBEXTRACT is called in order to initiate the typing of the extracted message.

### 3.7.9 Line Printer not Connected

The process of this message consists of resetting the flag HARDCOPY to 'FALSE', i.e. output is not directed to the line printer because it is not in operation.

4 Output Real-time Messages

In this section the format of the real-time messages sent by DB is described.

4.1 Synchronous Output

RMN : CS or LP

SMN : DB MN : 11

ML : depends on length of output text

DATAO: 0 - no roll screen/cr, lf after output

1 - roll screen/cr, If after output

DATA1-

DATAn: text to be typed/printed

This message is used to type text on the input line of the CRT or on the line printer.

4.2 CRT Input Request

RMN : CS

SMN : DB

MN : 12 ML : 05

DATAO: 0 - no roll screen after input

1 - roll screen after input

This message requests a keyboard input from the CRT.

4.3 Terminate CRT Input

RMN : CS

SMN : DB

MN : 14

ML : 04

This message disconnects DB from the CRT.

4.4 Activate Debug

RMN : CS

SMN : DB

MN : 15

ML : 04

This message is used to establish a connection between DB and the CRT at which the debug request was made.

#### 4.5 New Line

RMN : CS or LP SMN : DB MN : 16 ML : 04

This message rolls the CRT screen once and places the cursor at the beginning of the input line or positions the print head of the line printer at the beginning of the next line.

### 4.6 Begin of Line

RMN : CS or LP SMN : DB MN : 17 ML : 04

This message positions the CRT cursor at the beginning of the input line or the print head of the line printer at the beginning of the next line.

### 4.7 Synchronous Output with Output Acknowledge

RMN : CS or LP SMN : DB

MN : 18

ML : depends on length of output text

DATAO : acknowledge code

DATA1: 0 - no roll screen/cr, If after output 1 - roll screen/cr, If after output

-SATAG

DATAn: text to be typed/printed

This message is used to type text on the input line of the CRI or to print on the line printer and requests an acknowledge message when the output is completed.

### 4.8 External Start Debug

RMN : any external debug module

SMN : DB MN : 24 ML : 05

DATAO: module number of connected CRI module

This message informs an external debug module about a debug request from the CRT module determined by DATAO.

### 4.9 Start Message Extraction

RMN : EX SMN : DB MN : 10 ML : 04

This message informs the executive that a message extraction is started. The message to be extracted is described by the MCH currently in DEBUGMCH.

### 4.10 Stop Message Extraction

RMN : EX SMN : DB MN : 11 ML : 04

This message causes the termination of message extraction by the executive.

### 4.11 Initialize Line Printer

RMN : LP SMN : DB MN : 13 ML : 04

This message is sent when the output of DB is to be switched to the line printer.

# APPENDIX E

PROGRAM DESCRIPTION OF CRT MODULE

# TABLE OF CONTENTS

1	General	3
1.1	Introduction	3
1.2	Abbreviations and Conventions	3
1.3	Facilities of the CRT Module	4
1.3.2	Output	4
5	Inputs	6
2.1	Input from CRT	6
2.2	Input Real-time Messages	6
2.2.1	Start	6
2.2.2	Asynchronous Output	7
2.2.3	Synchronous Output	7
2.2.4	CRT Input Request	7
2.2.5	Activate CRT Input	7
2.2.6	Terminate CRT Input	8
2.2.7	Activate Debug	8
8.2.5	New Line	8
2.2.9	Beginning of Line	8
2.2.10	Synchronous Output with Acknowledge	9
3	Processing	0
3.1	USART Interface	0
3.2	Input from CRT	1
3.3	Output to CRT	1 3

3.4	Real-time Messages	14
3.4.1	Start	14
3.4.2	Asynchronous Output	14
3.4.3	Synchronous Output	15
3.4.4	CRI Input Request	15
3.4.5	Activate CRT Input	16
3.4.6	Terminate CRT Input	16
3.4.7	Activate Debug	16
3.4.8	New Line	16
3.4.9	Beginning of Line	16
3.4.10	Synchronous Output with Acknowledge	16
4	Outputs	1 7
4.1	Output to CRT	17
4.2	Output Real-time Messages	17
4.2.1	Transfer Input Text	17
4.2.2	Activate CRT Input Acknowledge	17
4.2.3	Terminate I/0	18
4.2.4	Start Debug	18
4.2.5	Terminate Debug Functions	18
4.2.6	Acknowledge	18

#### 1 General

### 1.1 Introduction

The CRT module (module identification: CS) is the driver for the CRT under the real-time operating system. All I/O is interrupt driven in order to meet the real-time

All I/O is interrupt driven in order to meet the real-time requirements.

The CS module 'connects' all other modules in the system to the CRT, i.e. the interface with the operator is controlled by the communicating module. The CS module merely passes data in and out for other modules. Except for some line editing functions CS does not have its own interface.

The interface between CS and a user module takes place with

The module may be shared by several SBC's. In this case only the message entry procedure and the CS data have to be local to the

SBC's.

Depending on the number of the host SBC the module numbers of the CS module vary.

CS is designed to communicate with a DATAMEDIA Elite 2500 CRT and keyboard.

# .2 Abbreviations and Conventions

All numbers in this segment of text are decimal except as indicated otherwise.

SBC - single board computer

the use of real-time messages.

MCB - message control block RMN - receiving module number

SMN - sending module number

MN - message number
ML - message length

CS - CRI module, module number in SBC 1/2/3: 06/14/22 EX - executive, module number in SBC 1/2/3: 00/08/16 DB - debug module, module number in SBC 1/2/3: 07/15/23

INACTMOD - module number of module currently connected to CRT (INput ACTive MODule)

input line - last line on CRT line for asynchronous outputs - line 7 on CRT line for synchronous cutputs - last line on CRT (=input line)

### 1.3 Facilities of the CS Module

### 1.3.1 Input

The CS module allows any module in the system to obtain data from an operator at the keyboard.

In order to ease keyboard input, CS provides two line editing functions:

- delete last input character
- delete entire input.

CRT input for a module is initiated with an 'activate CRT input' msq to CS. This msq is acknowledged with a msg sent back. The first data byte of this msg determines whether the request is acknowledged or not.

The case that CRT control is not granted can only occur if several modules attempt to obtain inputs at the same time. Once the connection to the CRT is established, the user module has to send an 'input request' msg to CS. CS then sends the next input terminated with the RETURN key to the requesting module.

All keyboard inputs are displayed on the input line.

Upon completion of the inputs, the connected module has to release the CRT with a special msg.

#### 1.3.2 Output

There are 2 distinct types of CRT output:

- asynchronous and
- synchronous.

Asynchronous outputs are typed on line 7 of the CRI. The text typed here is sent to CS using the 'asynchronous output' msg. This msg may be sent by any module at any time and serves the purpose to report errors, special internal conditions etc.

The output of this message is accompanied by the bell. For the time of the output all input is blocked and a 'f' is displayed at the position of the next input. After typing the asynchronous message the 'f' is erased with the next input.

Synchronous outputs are typed on the input line and allow the module currently in control of the CRT to interact with the operator.

'Synchronous output' messages are accepted from the connected module only.

There are two messages which control the screen itself. They cause the cursor to be positioned at

- the beginning of a new line (roll screen once)
- the beginning of the current line.

2 Inputs

The CS module receives inputs from the keyboard of the CRT and from other modules with real-time messages.

2.1 Input from CRT

Keyboard inputs are transmitted in form of ASCII characters. See DATAMEDIA Elite 2500 manual for details.

Apart from the normal character set there are inputs which have special meanings. They are listed below:

'Control=I' - CRI interruption
This input forces the termination of any
existing connection between a module and
the CRI.

'Control-D' - enter debug mode

'Control-I' - terminate debug function

'Back Space' - delete last character

'Rub Out' - delete entire input

'CR' - termination of current input

The interpretation of these inputs is described in Section 3.2.

2.2 Input Real-time Messages

In this section the format of the incoming real-time messages is given. The process of these messages is described in Section 3.4.

2.2.1 Start

RMN : CS SMN : EX MN : 00 ML : 04

This msg is sent by the executive when the system is started. Upon receipt the CS module initializes itself (see Section 3.4.1).

## 2.2.2 Asynchronous Output

RMN : CS

SMN : any module

MN : 10

ML : depends on length of text

DATAO-

DATAn : text to be typed

This message is received when a module is initiating an asynchronous output (see Section 3.4.2).

### 2.2.3 Synchronous Output

RMN : CS

SMN : module currently connected to CRI (INACTMOD)

MN : 11

ML : depends on length of text

DATAO: 0 - no roll screen after output

1 - roll screen after output

DATA1-

DATAn: text to be typed

The module currently connected to the CRT transmits text to be typed on the input line with this message (see Section 3.4.3).

#### .2.4 CRT Input Request

RMN : CS

SMN : INACTMOD

MN : 12 ML : 05

DATAO: 0 - no roll after next input

1 - roll screen once after next input

This message is sent by the module currently connected to the CRT to request an input from the keyboard (see Section 3.4.4).

# 2.2.5 Activate CRT Input

RMN : CS

SMN : module requesting connection to the CRT

MN : 13 ML : 04

With this message each module can establish a connection to the CRT (see Section 3.4.5).

### 2.2.6 Terminate CRT Input

RMN : CS

SMN : INACTMOD

MN : 14 ML : 04

This message is used by the module currently connected to the CRT to release the CRT (see Section 3.4.6).

## 2.2.7 Activate Debug

RMN : CS

SMN : any debug module

MN : 15 ML : 04

With this message the user selected debug module connects itself to the CRI (see Section.4.7).

#### 2.2.8 New Line

RMN : CS

SMN : INACTMOD

MN : 16 ML : 04

This message causes the roll of the screen by 1 line and the positioning of the cursor at the beginning of a new line (see Section 3.4.8).

#### 2.2.9 Begin of Line

RMN : CS

SMN : INACTMOD

MN : 17 ML : 04

This message positions the cursor at the beginning of the current input line (see Section 3.4.9).

# 2.2.10 Synchronous Output with Acknowledge

RMN : CS

SMN : INACIMOD

MN : 18

ML : depends on length of text

DATAO : acknowledge code

DATA1: 0 - no roll screen after output 1 - roll screen once after output

-SATAG

DATAn : text to be typed

This message initiates a synchronous output on the input line and requests an acknowledge message when this output is completed (see Section 3.4.10).

#### 3 Processing

In this section the process of CRT input/output and real-time message is described.

#### 3.1 USART Interface

The operation of the USART is controlled by a mode control word and a command control word.

After a hardware reset, the first output has to be a mode control word. After having received the mode control word the USARI accepts any number of command control words. If it is required to output a mode control word without a hardware reset, the USARI can be reset with a special control word.

The formats of the control words and the input/output ports are given below.

#### Mode Control Word:

" 3, 2: 10 - 7 BIT CHARACTER LENGTH

" 1, 0 : 10 - BAUD RATE FACTOR 16

#### Command Control Word:

bit 7: 0

" 6: 0 - NO RESET RESET USART

" 5 : 1 - REQUEST TO SEND

. 4: 0

" 3: 0

" 2: 1 - RECEIVE ENABLE

" 1: 1 - DATA TERMINAL READY

" 0 : 1 - TRANSMIT ENABLE

Input/Output port for status information is OEFH, for data OEEH.

#### 3.2 Input from CRT

All CRI input is interrupt driven.

Upon system start, the USARI (CRI input) is initialized and the respective interrupt is activated.

If a key on the keyboard is hit, an interrupt occurs. The interrupt procedure decodes the interrupt and schedules a call of the the priority procedure CSINPUT.

After the call of CSINPUT by the executive, an input instruction is executed to obtain the input character.

Without any filtering the following inputs are considered in the the indicated sequence and processed:

- 'Control-I' : CRT interruption
- 'Control-D' : initialize debug mode
- 'Control-I' : terminate debug function

# 'Control-I':

This input forces the termination of the current connection between a module and the CRT. If a module is currently connected to the CRT (INACTMOD <> FFH), then a 'terminate I/O' msg is sent to this module. If no module is connected, no actions take place.

## 'Control-D':

This input initiates the debug mode. If debugging from this CRT is already activated, the error indication '?' is written into the output buffer. If the request is legal, any current connection between a module and the CRT is terminated with the 'terminate I/O' message and a 'start debug' message is sent to the debug module in the same SBC.

#### 'Control-1':

This input causes the sending of a 'terminate debug function msg to the connected debug module. If not in debug mode, a '?' is written into the output buffer.

All other keyboard inputs are considered only if

- no output is active
- an 'input request' msq has been received.

#### 'Back Space':

This input causes the deletion of the last input character on the screen and in the input buffer.

The cursor is moved back and the last input is erased.

An attempt to move the cursor past the beginning of the last input request is not accepted and answered with the bell.

#### 'Rub Out':

With this key the entire input of the last input request can be erased. The cursor is moved back to the first position of the last input request and the rest of the line is deleted so that a correct input line can be generated.

#### 'RETURN':

This key terminates the current input.

The input collected starting at the position of the cursor at the last input request is sent to the connected module with a 'transfer input text' message.

Depending on the specification in the last 'input request' message, the screen is rolled once or not.

#### All other inputs:

The input ASCII character is checked for legality. Legal codes have to be within the range of 1fH < input code < 5AH and within the length of the input line. If the input is illegal a '?' is written into the output buffer, otherwise the input is stored, the input buffer pointers are updated and the input character is written into the output buffer. Before returning program control to the executive, the output of the current output buffer contents is initiated.

#### 3.2 Output to CRT

All CRT output is interrupt driven.

At system start the following codes are written into the output buffer:

- clear screen
- set roll mode
- output '\*\*\* SYSTEM START' and bell on the line for asynchronous outputs
- position cursor at the beginning of input line.

Then the output is activated by executing an output instruction for the first character in the output buffer. Upon completion of the output, an interrupt occurs which is processed by the interrupt routine. After determining the source of the interrupt, a priority call of CSOUTPUT is scheduled.

If there are no more characters for output, the output buffer pointers are reset to 0 and an acknowledge message is sent to INACTMOD if requested.

Otherwise the next character from the output buffer is transferred to the USARI and the buffer pointers are updated.

The output buffer CRIOUT has 2 pointers:

- OUIPTR (next character for output)
- OUTX (next character to fill).

CRIOUT is empty if OUTPIR >= OUTX. In this case both pointers are reset to 0.

# 3.4 Real-time Messages

#### .4.1 Start

This message informs CS that the system has been started. CS performs the following initialization steps:

- reset USART if no system reset has been performed
- clear all local variable data
- determine begin addresses of priority procedures CSINPUT and CSOUTPUT and store in CSINADR and CSOUTADR respectively
- set module status:
  - . module exists
  - . module activated
  - module authorized to suspend/activate other modules
- reset pointers and flags
- enter priority calls of CSINPUI and CSOUIPUI
- initialize USART
- pack 'start output' into output buffer
- initialize output to USART

#### 3.4.2 Asynchronous Output

With this messages text to be typed on the line for asynchronous outputs (line 7 on CRT) is sent to CS.

For the duration of the output, an uparrow ('î') is displayed at the current position of the cursor on the input line.

After completion of the asynchronous output, the cursor is returned to its last position on the input line. The next keyboard input erases the '1'.

The following output is initiated:

- '1'
- move cursor to the beginning of line 7
- clear line
- pack bell
- pack '\*\*\* '
- pack output text (from message)
- pack code to return cursor to the last input position.

After packing the output buffer, the output is initiated with the call of STARTOUT.

#### 3.4.3 Synchronous Output

This message causes the typing of the transferred text on the input line.

If SMN of this message is different from INACTMOD, then the system routine ILLEGA is called and the message rejected.

If the message is legal, the text of the message starting at data byte 1 is moved into the output buffer.

If data byte 0 is TRUE, then the roll of the screen after the output is requested and the respective characters are entered into the output buffer.

If no roll is requested, the position of the cursor on the input line is incremented by the number of characters in the transferred text.

The output to the CRT is initiated with a call of procedure STARTOUI.

# 3.4.4 CRT Input Request

This message requests an input from the CRI keyboard.

If SMN is not INACIMOD, then the system routine ILLEGALMSG is called and the request is rejected.

The cursor position and the input buffer pointer are saved in order to mark the beginning of this input for line editing.

The variable ROLLSCREEN is set according to data byte 0 of the msg.

#### 3.4.5 Activate CRT Input

This message is sent by a module which wishes to establish a connection to the CRT.

This request is answered with an acknowledge message in which the first data byte indicates whether the request is acknowledged or not.

SMN of this message is saved in the variable INACTMOD.

#### 3.4.6 Terminate CRT Input

This message terminates the connection between the sending module (SMN = INACIMOD) and the CRT.

CS resets INACIMOD to FFH.

#### 3.4.7 Activate Debug

This message connects a debug module to the CRT. No acknowledge message is sent in this case since this request is always granted.

INACIMOD is set to the module number of the debug module.

#### 3.4.8 New Line

This message is internally converted to a 'synchronous output' message and the procedure for synchronous output is called.

#### 3.4.9 Beginning of Line

See Section 3.4.8

#### 3.4.10 Synchronous Output with Acknowledge

The process of this message is essentially identical to the process of a synchronous message without acknowledge (see Section 3.4.3) with the exception that the sending module requests a message to indicate the completion of the initiated output.

This feature can be used to avoid overwriting of text and buffer overflows.

The first data byte contains the acknowledge code which has to be returned with an acknowledge message after completion of the output.

#### 4 Outputs

The CS module outputs data to the CRT and sends real-time messages to other modules in the system.

# 4.1 Output to CRT

Outputs to the CRI are transmitted in form of ASCII characters. See DATAMEDIA Elite 2500 manual for details.

The only ASCII exception is the use of the XY addressing feature of the DATAMEDIA. This mode allows the arbitrary positioning of the cursor. The XY mode is initiated with 'OCH'. The following two characters (coded according to the manual) are considered to be a location on the CRI. The first determines the position in a row and the second character determines the row.

If CS detects an illegal input character, then this is indicated with the output of '?'.

#### 4.2 Real-time Messages

In this section the format of the real-time msg sent by CS is described.

#### 4.2.1 Transfer Input Text

RMN : INACTMOD

SMN : CS MN : 20

ML : depends on length of input text

DATAO-

DATAn : input text

This message is used to send input text to the requesting module.

#### 4.2.2 Activate CRT Input Acknowledge

RMN : INACTMOD

SMN : CS MN : 21 ML : 05

DATAO: IRUE - request acknowledged

FALSE - otherwise

This message informs the module which wants to activate CRI input whether the request is granted or not.

#### 4.2.3 Terminate I/O

RMN : INACTMOD

SMN : CS MN : 22 ML : 04

This message is sent to the module currently in control of the CRT when the CRT interruption input (Control-I) is detected.

#### 4.2.4 Start Debug

RMN : DB SMN : CS MN : 23 ML : 04

This msg informs the debug module that the legal input 'Control-D' has been generated.

# 4.2.5 Terminate Debug Function

RMN : DB SMN : CS MN : 25 ML : 04

This message informs the debug module that the input of 'Control-I' has been detected.

### 4.2.5 Acknowledge

RMN : INACTMOD

SMN : CS MN : 26 ML : 05

DATAO : acknowledge code

This message indicates the completion of an output with acknowledge request.

The acknowledge code is identical to the code received with the 'synchronous output with acknowledge' message.

# APPENDIX F

PROGRAM DESCRIPTION OF LINE PRINTER MODULE

# TABLE OF CONTENTS

1	General	3
1.1	Introduction	3
1.2	Abbreviations and conventions	3
1.3	Facilities of the LP module	4
5	Inputs	4
2.1	Status Inputs from Line Printer	4
2.2	Input Real-time Messages	5
2.2.1	Start	5
2.2.2	Print	5
2.2.3	New Page	5
2.2.4	Initialize Line Printer	5
2.2.5	New Line	5
2.2.6	Beginning of Line	6
2.2.7	Print with Acknowledge	6
3	Processing	7
3.1	Line Printer Interface	7
3.2	Line Printer Status Inputs	7
3.3	Line Printer Output	7
3.4	Real-time Messages	8
3.4.1	Start	8
3.4.2	Print	8
3.4.3	New Page	8
3.4.4	Initialize Line Printer	8
3.4.5	New Line	8
3.4.6	Print with Acknowledge	9

4	Outputs	10
4.1	Output to Line Printer	10
4.2	Output Realtime Messages	10
4.2.1	Asynchronous Output	10
4.2.2	Acknowledge	10
4.2.3	Line Printer not Connected	10

#### 1 General

#### .1 Introduction

The line printer module (LP) is the driver for the CENTRONIX 101 line printer under the real-time operating system.

All output is interrupt driven in order to meet the real-time requirements.

The code of the LP module may be shared by several single board computers if more than 1 line printer is to be connected to the system.

All modules in the system can initiate printouts on the line printer by transferring text to LP with a real-time message.

## 1.2 Abbreviations and Conventions

All numbers in this segment of text are decimal except when indicated otherwise.

SBC - single board computer

MCB - message control block

RMN - receiving module number

SMN - sending module number

MN - message number

ML - message length

CS - CRT module, module number in SBC 1/2/3: 06/14/22

EX - executive, module number in SBC 1/2/3: 00/08/16

DB - debug module, module number in SBC 1/2/3: 07/15/23

LP - line printer module, module number in SBC 1/2/3: 05/15/21

# 1.3 Facilities of the LP Module

In order to allow the switching of output between CRT and line printer, the usage of the line printer is very similar to the CRT output, i.e. is controlled by the same messages. (Exception: the 'beginning of line' message is interpreted as 'new line' message.)

LP allows to print any text sent to the module with the 'print' message. The text is transferred in the data bytes of the message and has to be ASCII coded. This text can be printed with or without the generation of a acknowledge msg to indicate the completion of the printout.

Apart from the 'print' and 'print with acknowledge' message there are 3 form control message:

- new page
- new line
- beginning of line (interpreted as new line).

LP accepts an 'initialize line printer' message. Upon receipt of this message the proper connection of the line printer is checked.

If the check is positive, the print head is positioned at the beginning of a new page.

Otherwise a warning message is typed on the CRT and a 'line printer not connected' message is sent to the module which requested the check.

#### 2 Inputs

The LP module receives inputs from

- line printer (status inputs)
- other modules (real-time messages).

#### 2.1 Status Inputs from Line Printer

Two status bits from the line printer are made available for LP on input port E6:

- bit 4 : 'SELECT'
- bit 3 : 'LPT BUSY'.

### 2.2 Input Real-time Messages

In this section the format of the incoming real-time message is given.

The process of these messages is described in Section 3.4.

#### 2.2.1 Start

RMN : LP SMN : EX MN : 00 ML : 04

This message is sent by the executive when the system has been started.

Upon receipt the LP module initializes itself (see Section 3.4.1).

#### 2.2.2 Print

RMN : LP

SMN : any module

MN : 11

ML : depends on length of text to be printed

DATAO : 0 - no new line after output

1 - new line after output

DATA1-

DATAn : ASCII characters to be printed

The text transferred with this message is to be printed on the line printer (see Section 3.4.2).

#### 2.2.3 New Page

RMN : LP

SMN : any module

MN : 12 ML : 04

This message requests the positioning of the print head at the beginning of the first line on a new page (see Section 3.4.3).

#### 2.2.4 Initialize Line Printer

RMN : LP

SMN : any module

MN : 13 ML : 04

This message causes LP to check the status of the line printer (see Section 3.4.4).

#### 2.2.5 New Line

RMN : LP

SMN : any module

MN : 16 ML : 04

This message positions the print head at the beginning of a new line (see Section 3.4.5).

#### 2.2.6 Beginning of Line

RMN : LP

SMN : any module

MN : 17 ML : 04

This message is processed as being a 'new line' msg (see Section 2.2.5).

## 2.2.7 Print with Acknowledge

RMN : LP

SMN : any module

MN : 18

ML : depends on length of text

DATAO: acknowledge code

DATA1: 0 - no new line after print

1 - new line after print

-SATAG

DATAn: ASCII characters to be printed

The transferred text of this message is to be printed and upon completion of the printing an acknowledge message with DATAO has to be returned to the sending module (see Section 3.4.6).

- 3 Processing
- .1 Line Printer Interface
  - 3.2 Line Printer Status Input

Upon receipt of the 'initialize line printer' message, LP reads the 'SELECI' status bit of the line printer.

If the line printer is connected ('SELECI' bit = 'FALSE') the procedure LPNEWPAGE is called.

If the line printer is not connected properly:

- an 'asynchronous output' msg is sent to CS to type a warning '\*\*\* LINE PRINTER NOT CONNECTED'
- a 'line printer not connected' msg is sent to the module which sent the 'initialize line printer' msg.
- 3.3 Line Printer Output

All line printer output is interrupt driven.

The output characters are written into the output buffer LPOUTBUF. LPOUTBUF is controlled by two pointers:

- LPOUTX (next to fill)
- LPOUTPIR (next to print).

LPOUTBUF is empty if LPOUIPIR >= LPOUTX. In this case both pointers are reset to 0.

An overflow occurs if LPOUIX > last index of LPOUIBUF. This condition causes a call of the system monitor and output characters are rejected until there is space in LPOUIBUF.

When all text is transferred from the 'print' message into the output buffer, or an overflow occurs, the output is started by calling the output procedure.

When the output of a character is completed, an interrupt is generated and the priority call of the procedure LPOUTPUT is scheduled by the interrupt procedure.

After LPOUTPUT called by the executive, the next character is is put on the output line or the printout is terminated (if LPOUTBUF is empty).

In the latter case the flag LPTELLBACK is checked and an acknowledge message is generated if necessary.

#### 3.4 Real-time Messages

# .4.1 Start

This message informs LP that the system has been started. LP performs the following operations:

- clear all local variables
- determine start address of procedure LPOUIPUI
- enter priority call for LPOUTPUT (to be called if a line printer output interrupt occurs)
- set module status
  - . module exists
  - . module activated.

#### 3.4.2 Print

This message is processed in procedure LPPRINT.

The data bytes of the msg are moved into the output buffer until all characters are moved or an overflow occurs. In case of an overflow, the move of characters is terminated and the output is initiated.

When the remaining space is sufficient and all characters are moved, then DATAO of the msg controls the insertion of code for a 'carriage return' and a 'line feed'. Then the output is initiated.

#### 3.4.3 New Page

The code for 'form feed' is packed into the output buffer and the output is activated.

#### 3.4.4 Initialize Line Printer

LP reads the 'SELECT' status bit and determines whether output is possible or not.

If no printing is possible, i.e. the printer is not connected properly, the typing of a warning message at the CRT is generated with an 'asynchronous output' msg to CS.

Otherwise, the 'new page' procedure is executed (see Section 3.4.3).

#### 3.4.5 New Line

The code for 'carriage return' and 'line feed' is packed into the output buffer and the output is activated.

## 3.4.6 Print with Acknowledge

This message is processed by the procedure LPPRINT (see 3.4.2) after the acknowledge code has been removed from the msg. The code and the sending module is saved. After completion of the output an acknowledge message is sent back together with the requested acknowledge code.

#### 4 Outputs

The LP module outputs data to the line printer and sends messages to other modules in the system.

#### Output to the Line Printer 4.1

Outputs to the line printer are transmitted in form of ASCII characters.

Letters are printed as capital letters only.

See the available CENTRONIX 101 interface specification.

Output port is port E4H.

#### 4.2 Real-time Messages

#### 4.2.1 Asynchronous Output

RMN : CS SMN LP : MN 10 : 30 ML

DATAO-

DATA25: text 'LINE PRINTER NOT CONNECTED'

#### 4.2.2 Acknowledge

module which sent 'print with acknowledge' message RMN

SMN LP : MN 26 : 05 ML

acknowledge code transferred with 'print with DATAO : acknowledge' message

#### 4.2.3 Line Printer not Connected

module which sent 'initialize line printer' message RMN

SMN LP MN 28 : ML 04

This message informs the requesting module that the line printer is not connected properly to the system.

# APPENDIX G

PROGRAM LISTINGS

EXECUTIVE

```
DECLARATIONS OF MSG ENTRY PROCEDURES FOR ALL POSSIBLE
                                                                                                         IF A MODULE IS ADDED THE RESPECTIVE MSGENT PROCEDURE
                                                                                                                      HAS TO BE MADE 'EXTERNAL' AND THE 'RETURN' HAS TO
                                                                                                                                                                                                                                                                                                                                                      MSGENTØS: PROCEDURE EXTERNAL;
                                                                                                                                                                                                                                                                                                                                                                                 MSGENTOG: PROCEDURE EXTERNAL;
                                                                                                                                                                                                                                                                                                                                                                                                           MSGENT07: PROCEDURE EXTERNAL;
              # INCLUDE(:F1:MSGENT, SRC)
                                                                                             MODULES IN COMPUTER
                                                                                                                                                                                        MSGENT 81: PROCEDURE;
                                                                                                                                                                                                                                                                                                               MSGENT 64: PROCEDURE;
                                                                                                                                                                                                                               MSGENT02: PROCEDURE:
                                                                                                                                                                                                                                                                     MSGENTØ3: PROCEDURE;
                                                                                                                                      BE TAKEN OUT
                                                                                                                                                                                                      RETURN
                                                                                                                                                                                                                                             RETURN
                                                                                                                                                                                                                                                                                                                           RETURN
                                                                                                                                                                                                                                                                                     RETURN
                                                                                                                                                                                                                                                                                                                                           END;
                                                                                                                                                                                                                     END;
                                                                                                                                                                                                                                                            END
                                                                                                                                                                                                                                                                                                    END;
$EJECT
                                                                                                                                       11
                                                                                                                                                                                                                                 122
123
124
126
126
128
128
129
138
131
                                                                                                                                                                                                                    121
                                                                                                                                                                                                                                                                                                                                                                   133
133
134
134
```

# PL/M-80 COMPILER

\* EJECT

EXTERNAL	EXTERNAL;
: PROCEDURE EXTERNAL; END EXMSGENT;	PROCEDURE EXTERNAL,
EXMSGENT: END	EXSTART:
40	-

137 138

EXTERNAL;	EXTERNAL
PROCEDURE EXTERNAL END EXSTART;	TR: PROCEDURE END EXMSGEXTR:
EXSTART: END	EXMSGEXTR: PROCEDURE EXTERNAL END EXMSGEXTR:
ત્તા	40
139 1	141 1

*/	
*********	*************************************
********	*************************************
**	

	VE		· · · · · · · · · · · · · · · · · · ·
**	EXECUTIV	**	· · · · · · · · · · · · · · · · · · ·

**************************************
--

**************************************		
********	SHINDERS PRINTER THERE OF MODIFIES	
*******	TASKS OF	
******	PRICETTY	
******	SSECUER	
***	*	*

EXEC1: DO WHILE PRIORSCHEDULE <> 0;	/* DO AS LONG AS PRIORITY CALLS ARE SCHEDULET	XB2 = 1;	50 XB1 = 1 TO MAXPETOP:
EXE			
N		M	N
145 2		146 3	147

# PL/M-80 COMPILER

/* FIND HIGHEST PRIORITY SCHEDULED */ XB3 = SCR(PRIORSCHEDULE, XB1); IF CARRY THEN	PRIORSCHEDULE = PRIORSCHEDULE XOR XB2; /* RESET PRIORITY BIT */	XA1 = PRIORLIST(XB1-1);	CALL XA1;  /* CALL PRIORITY PROCEDURE */	GO TO EXEC11;	END;	xB2 = ROL(xB2,1)	END;	EXEC11: ,	END;	**	*************************************	**	PROCESS PENDING REAL TIME MESSAGES	*	EXEC2: IF NUMMSG = 0 THEN GO TO EXEC21;	/* NO MSG TO PROCESS */	IF MSGOUT = LASTMSG THEN LASTMSG, MSGOUT = 0;	/* NEXT MSG AT TOP OF MSGBUFFER */	IF MSGEXTRACTION THEN CALL EXMSGEXTR:	VA TOO EXTRACTION 15 DOLLANDED 47	XH1 = MSG001)	MSGOUT = XA1 + MSGBUFFER(XA1+ML); /* COMPUTE BEGIN OF NEXT MSG */	NUMMSG = NUMMSG - 1;	OCCUMENT - MODERNING - MSG +/	HORMOG = . MOGBOFFER(XHLX)
444	r ID	ທ	ທ	n	ທ	4	4	M	M						N		7		N	(	V	N	N	c	V
44 8 44 8 49	151	152	153	154	155	156	157	158	159						169		162		164	100	100	167	168	160	163

œ
1LER
=
Ť
COMP
o
88
1
ŧ
Ĺ

ADRNSGDATA = ADRMSG + 4; /* 'SET' MSG INTO WORKAREA */	XB1 = MSG(RMN) - FIRSTMN; /* COMPUTE INTERNAL MODULE NUMBER */	IF XB1 >= MAXMOD THEN XB2 = SENDEXT;	/* MODULE NOT IN OWN COMPUTER CALL EXTERNAL COMMUNICATION */	ELSE DO CASE XB1;	CALL EXMSGENT;	CALL MSGENTØ1;	CALL MSGENT02;	CALL MSGENTØ3;	CALL MSGENTØ4;	CALL NSGENTØ5;	CALL MSGENTØ6;	CALL MSGENTØ7;	END;	GO TO EXEC1;	/* CHECK PRIORITY CALLS AGAIN */	EXEC21: IF EXIMSG <> CPTR THEN GOTO EXEC3; /* NO EXTERNAL MSG TO PROCESS */	CALL RECEXT,	/* TRANSFER MSG INTO OWN MSG BUFFER */	GO TO EXEC1;	/* CHECK PRIORITY CALLS AGAIN */	**	***************************************	CHECK FOR PERIODIC CHLLS OF MODULES	EXEC3:	
N	N	N		N	M	M	M	M	M	M	M	M	M	N		N	Ø		N					N	
178	171	172		174	175	176	177	178	179	180	181	182	183	184		185	187		188					189	

The state of the s

IF NUMPER = 0 THEN GO TO EXEC4; /* NO PERIODIC CALLS ACTIVE */	XB1 = NUMPER - 1; /* SET LIMIT FOR 1 SEARCH CYCLE */	DO XB2 = 0 TO XB1; /* CHECK ALL ACTIVATED PERIODICS */	PERX = PERXTBL(XB2);	IF TIMECHKS, PERLISTSPERX), PERTIME(0)) THEN	CALL TIME KEHCHED */		CHLL SETPERTIMECPERXXX  /* SET NEXT CALL TIME */	XA1 = PERLIST(PERX), PERRDR;	CRLL XR1;	/* CALL PERIODIC */	GO TO EXEC1;	/* CHECK PRIORITY CALLS */	END;	EMD;	*	*************************************	**	BACKGROUND TASKS	/*	EXEC4: ,	GO TO EXEC1;	RETURN	END EXEC;	
N	N	N	M	M	٨	1	4	4	4		4		4	M						N	N	N	N	
196	192	193	194	195	100	100	197	198	199		208		261	202						203	284	205	286	

# PL/M-80 COMPILER

MAIN MODULE PROGRAM ENTRY

287

208

\*

SYSSTART: SAVESTACKPTR = STACKPTR;

/\* SAVE STACKPOINTER FOR RESTART \*/
RESTART = FALSE;

/\* START WITH SYSTEM RESET \*/

CALL EXEC;

END

203

MODULE INFORMATION:

461D 747D 4D VARIABLE AREA SIZE = 02EBH MAXIMUM STACK SIZE = 0004H 705 LINES READ 0 PROGRAM ERROR(S) = 61CDH CODE AREA SIZE

END OF PL/M-80 COMPILATION

MESSAGE HANDLER OF EXECUTIVE

ISIS-II PL/M-80 V3.0 COMPILATION OF MODULE EXMSG NO OBJECT MODULE REQUESTED COMPILER INVOKED BY: PLM80 :F1:EXMSG.SRC NOOBJEC

PLM80 :F1:EXMSG. SRC NOOBJECT PAGEWIDTH(80) PAGELENGTH(35)

ĝ EXMSG: \* ત

UPDATE: 1

\* NOLIST

EXEC MSG HANDLER

PROCEDURE EXTERNAL;

END EXEC; EXEC HN 117

PROCEDURE EXTERNAL MSGENT07: HN 119

 EXPER

N

0

# THE GO COMPILER

# EXECUTIVE PERIODIC

```
EXMSGEXTR
                                                                                                                                                         /* SEND MSG OVERFLOW TO SYSTEM MONITOR */
                                IF EXPERFL = 0 THEN
/* FIRST CALL, GET ADDRESS */
                                                                                           CALL SYSMON(CODESAVE, 0, 1);
                                                                                                            CALL PERSUSP(EXPERX);
                                                         EXPERADR = PROCADR;
                                                                  EXPERFL = 1;
                                                                                                                     EXPERX = OFFH;
               PROCEDURE;
                                                                                                                                       END EXPER:
                                                                           RETURN
                                                                                                                              RETURN
                                                                                    END
               EXPER:
                                                                                                                                                                          **
                                                                                                             2000
                                                                                                            2525
                                                 248848
```

170

EXTRACT CURRENT MSG IF REQUIRED

1 EXMSGEXTR: PROCEDURE PUBLIC:

ADRNSG = . MSGBUFFER(MSGOUT);

Ø

134

M

DO XB3 = 0 TO 3; /* CHECK MCB OF CURRENT MSG */	IF (DEBUGNCB(XB3) <> OFFH) AND (DEBUGNCB(XB3) <> MSG(XB3)) THEN RETURN: /* NO EXTRACTION REQUIRED */	END	BATKMSG(ML) = MSG(ML) + 4)	ADRMSGDATA = . MSGBUFFER(MSGOUT);	CALL MSGENTØ7;	RETURN	END;	*	**************************************	춙쏡춖춖춖춖춖춖춖춖춖춖춖춖춖춖춖춖춖춖춖춖춖춖춖춖춖춖춖춖춖춖춖춖춖춖춖춖춖	**	EXMSGENT	PROCESS REAL TIME MESSAGE	**	老者我不会在本人的人的人,我们也是一个人的人的人的人的人,我们们的人们的人们的人们的人们的人们的人们的人们的人们的人们的人们的人们的人们的人	*	EXMSGENT: PROCEDURE PUBLIC;	IF MSG(MN) > 9 THEN	DQ;	XB3 = MSG(MN) - 16;	IF XB3 > 1 THEN RETURN.	DO CASE XB3;	MSGEXTRACTION = TRUE;	MSGEXTRACTION = FALSE,	END;	EAC
N	M	M	NO	10	N	N	N										7	N	N	M	M	M	4	4	41	M
135	136	138	139	141	142	143	144										145	146	147	148	149	151	152	153	154	122

		ı
i	١	į

	,
0	2
9	4
9	4
9	K
9	K
00	LER
22.	LER
03 1	LER
1	TIEN
1	TILER
1	TILER
1	LILER
1	IL TLEK
1	IL TEEK
1	ILL TLEK
1	JUL 1 LEK
1	OFF LEEK
1	OUT LEEK
COMPTI ED	TOTAL TEEN
1	COLL LLER
1	COLL LEEK
1	COLLER
1	COLLICER
COMPT	5
1	5
COMPT	5
LOG COMPT	100
LOG COMPT	5
LOG COMPT	100
MASS COMPTI	100
LOG COMPT	100

•															
LAM-68 CONTILER	2	SYSTEM RESTART ** ********************************	2 DCL (START BASED A1) BYTE;	2 IF RSTFL = 0 THEN	2 DO; /* FIRST CALL, FIND ADDRESS OF SYSRESTART */	RSTA	3 RSTFL = 1;	3 RETURNS	3 END;	2 IF CPTR $\circlearrowleft$ 1 THEN		3 IF CPTR = 2		3 DO WHILE START <> 0; /* WAIT UNITE SEC 1 STARTED */	4 END;
0															,
	156	158	159	168	161	162	163	164	165	166	167	168	178	171	172

o
ILER
COMP
8
Ø
-86
Ė
ž

START = CPTR; END;	RESTART = TRUE; /* START WITHOUT RESET */	STACKPTR = SAVESTACKPTR; /* RESET STACK POINTER */	CALL EXEC;	END SYSRESTART; /* **********************************	*** START	INITIALIZE COMPUTER ** *********************************	*/ EXSTART: PROCEDURE PUBLIC;	DISABLE; CALL CLEARDATAC.EXCLEARBEG, EXCLEAREND); /* CLEAR EXEC DATA */	DO B1 = 0 TO LAST(PERLIST), /* INITIALIZE PERIODIC   151 */	PERLIST(81). FREE = TRUE; END; ADRSTAT = . MODSTATUS(8) + FIRSTMN;
мм	N	a	N	a			4	NN	N	mma
173	175	176	177	178			179	180	182	183 184 185

ω

/* LOCATE OVERLAY FOR HOST COMPUTER IN MODSTATUS TABLE */	MODSTAT(0) = 00001011B; /* SET EXEC SLOT */	DO B1 = 1 TO LAST(MODSTAT);  /* INITIALIZE MODSTAT WITH 0  I. E. NO MODULE EXISTS PACK INTERNAL START MSG INTO MSGBUFFER */	MODSTAT(B1) = 0;	SYSNNB(0) = B1 + FIRSTNNJ	CALL PACKMCB(. SYSMNB(0))	END;	NUMMSG = MAXMOD - 1; /* SET NUMBER OF MSG IN MSGBUFFER */	CALL EXPER; /* OBTAIN ADDRESS OF EXPER */	EXPERX = OFFH,	A4 = INTVECTOR;	B1 = BU AND 11100000B;	OUTPUT(GDAH) = B1 OR 00010010B; /* ICW1 */	OUTPUT(0DBH) = BL; /* ICW2 */ /* INITIATE INTERRUPT CONTROLLER, FULLY NESTED MODE */	IF CPTR = 1 THEN INTMASK = 01111001B;	ELSE INTMASK = 01111101B;	OUTPUT(BOBH) = INTMASK;  /* INITIALIZE INTERRUPTS  INT? - SYSTEM START  INT2 - RTC (SBC 1 ONLY)	
	N	N	M	M	M	M	N	N	N	N	N	N	N	N	N	N	
	186	187	188	183	190	191	192	193	194	195	196	197	198	199	201	202	

~

#### PL/M-80 COMPILER

INT1 - ENTER MONITOR (FRONT PANEL INT2)

OUTPUT(8DFH) = 001100008;
/\* COUNTER 0 MODE CONTROL WORD \*/

OUTPUT (BDCH) = 1AH;

NN

284

283

/\* LOAD COUNTER 0 (RTC), LEAST SIGNIFICANT BYTE FIRST 1 MSEC = 538 X 1860 NSEC <=> 21AH \*/ OUTPUT(@DCH) = 2,

IF CPTR = 1 THEN

CALL CLEARDATA( BEGABSDATA, ENDABSDATA);

IF RESTART THEN DO;

START2 =

ö ö START3

END

ELSE DO; START2 = 2; START3 = 3;

/\* START SBC2 AND SBC3 \*/

ENABLE;

N

219

218

EMD;

END EXSTART; RETURN 204

END EXMSG:

222 221 222

ત

PL/M-80 COMPILER

ISIS-II PL/M-80 V3.0 COMPILATION OF MODULE INT NO OBJECT MODULE REQUESTED COMPILER INVOKED BY: PLMS0 :F1:INT. SRC NOOBJE

PLM80 :F1:INT. SRC NOOBJECT PRGEWIDTH(80) PRGELENGTH(35)

ĝ \* NOLIST

INTERRUPT PROCESSING

OUTPUT(8DRH) = 661666668;PROCEDURE INTRESET 118 117

END INTRESET; RETURN NNN

119

PROCEDURE INTERRUPT 6; IMTB: 121

/\* CDC INTERRUPT \*/

IF NOT CDCACTIVE THEN RETURN DISABLE NNN

122 124 125

CALL CDCRDR;

/\* CALL PROCESS OF INTERRUPT \*/

œ
W
ILER
=
COMP
O
0
8
1
Ė
Ą

INTMASK = INTMASK OR 1; OUTPUT(0DBH) = INTMASK; /* DISABLE INT0 */	CALL INTRESET; CDCACTIVE = FALSE;	ENABLE; RETURN;	END INTO:	/* ***********************************	/* RTC INTERRUPT */	DCL INCRONE BYTE DATA (1);	DISABLE	RTC(3) = RTC(3) + INCRONE; RTC(2) = RTC(2) PLUS 0; RTC(1) = RTC(1) PLUS 0; RTC(0) = RTC(0) PLUS 0; /* UPDRTE RTC */	OUTPUT(0DFH) = 00110000B; OUTPUT(0DCH) = 1AH; OUTPUT(0DCH) = 2; /* SET CTR 0 TO 1 MSEC IN MODE 0 */	CALL INTRESET; ENABLE;
00	aa	0 0	N	4		0	N	0000	000	NN
126 127	128	130	132	133		134	135	136 137 138 139	140 141 142	143 144

OW

145

/\* SCHEDULE PRIORITY CALL \*/ CALL PRIORITYINT(INTTBL(3>) PROCEDURE INTERRUPT 3; DISABLE ENABLE; INT3: NNNN OO 148 149 158 151 147 152

END INT3; RETURN

PROCEDURE INTERRUPT 4; INT4: 154

/\* SCHEDULE PRIORITY CALL \*/ CALL PRIORITYINT (INTTBL(4)); DISABLE ON 155 156

CALL INTRESET; END INT4; ENABLE; RETURN NNNN 157 158 159 160 \* INTS

161

PROCEDURE INTERRUPT 5,

This page intentionally blank

### PL/M-80 COMPILER

DISABLE	CALL PRIORITYINT(INTTBL(5));	CALL INTRESET;	ENABLE	RETURN	END INTS;	*	**************************************	/*	END INT;	
N	N	N	N	N	N				<del>,</del>	
162	163	164	165	166	167				168	

# MODULE INFORMATION:

173D	90	100		
ВВЯБН	ВВВВН	врен		
11	Ħ	H		
CODE AREA SIZE	VARIABLE AREA SIZE	MAXIMUM STACK SIZE	571 LINES READ	0 PROGRAM ERROR(S)

# END OF PL/M-80 COMPILATION

PUBLIC DECLARATIONS OF SYSTEM CALLS

ન

# PL/M-80 COMPILER

PLM80 :F1:SCPUB1. SRC NOOBJECT PAGEWIDTH(80) PAGELENGTH(35) ISIS-II PL/M-80 V3. 0 COMPILATION OF MODULE SCPUB1 NO OBJECT MODULE REQUESTED COMPILER INVOKED BY: PLMSQ

ĝ \$ NOLIST SCPUB1: 4

PROCEDURE (P1, P2, P3) EXTERNAL; (P2, P3) BYTE, P1 ADDRESS, END SYSMON; SYSMON: DCL 400 848

\*

 CLEARDATA

CLEAR MODULES VARIABLE DATA REGION

INPUT : P1 - ADDRESS OF FIRST BYTE P2 - ADDRESS OF LAST BYTE

PROCEDURE (P1. P2) PUBLIC;

CLEARDATA

23

(P1, P2) ADDRESS, (I, J) ADDRESS, (DAT BASED P1) (1) BYTE, g O 34

I = P2 - P1; D0 J = 0 T0 I; ON 88 N

DAT(J) = 0; END; RETURN; END CLEARDATA;	**************************************	RETURN BYTE WITH A 1 SHIFTED INTO POSITION P1  INPUT : P1 - BIT POSITION  OUTPUT : P1TH POWER OF 2  ** ********************************	*/ SHFT: PROCEDURE (P1) BYTE PUBLIC;	DCL P1 BYTE;	IF P1 = 0 THEN RETURN 1; RETURN ROL(1, P1); END;	*/
MMNN			त्त	N	010101	
78 88 88 89 89			41	4	± 4 4 4 €	

CHECK A BIT IN A SPECIFIED BYTE INPUT : P1 - BIT P2 - BYTE TO CHECK

BIT

NAVAL POSTGRADUATE SCHOOL MONTEREY CALIF
A REAL-TIME OPERATING SYSTEM FOR SINGLE BOARD COMPUTER BASED DI--ETC(U) AD-A059 601 JUN 78 W NIEMANN NL UNCLASSIFIED 30=4 AD59 601 Ø - Ningh L. HA

M

# PL/M-80 CONPILER

MOT
H
FALSE
SET,
15
BIT
IF
TRUE
-
OUTPUT

CLEAR A BIT IN A SPECIFIED BYTE INPUT : P1 - BIT P2 - ADDRESS OF BYTE

PROCEDURE (P1, P2) PUBLIC; P1 BYTE, P2 ADDRESS, (WORD BASED P2) BYTE, CLEARBIT: δ Z N 23 45

XB1 = SHFT(P1); WORD = (WORD AND XB1) XOR XB1; /\* CLEAR BIT \*/ 20 8 S

a.
1LER
F
포
SQMF
_
8
T
Ė
ş
u.

RETURN; END CLEARBIT; /* **********************************	SET A BIT IN A SPECIFIED BYTE  INPUT : P1 - BIT  P2 - ADDRESS OF BYTE  ** ********************************	SETBIT: PROCEDURE (P1, P2) PUBLIC;	DCL P1 BYTE, P2 ADDRESS, (WORD BASED P2) BYTE;	WORD = WORD OR SHFT(P1); /* SFT RIT */	RETURN; END SETBIT; /* **********************************
NN		4	N	N	NN
% %		53	89	61	63 62

CHECK P1 AGAINST P2 INPUT : P1,P2 VALUES TO CHECK P3 ERROR CODE FOR SYSMON

LEGAL

Ŋ

OUTPUT : TRUE IF P1 < P2 FALSE OTHERWISE

	******	
	**************************************	
	***********	
**	********	1

PUBLIC:	
BYTE	
(P1, P2, P3) BYTE PUBLIC	Ē
PROCEDURE	(P1, P2, P3) BYTE
LEGAL:	정
4	N
64	65 2

(FL) FE, FS) BTIE	IF P1 < P2 THEN RETURN TRUE;	CRLL SYCMOMESTREKPTR, 8, P30;
7		
v	N	0
2	66 2	00

CALL SYSMON(S' RETURN FALSE; END LEGAL; END SCPUB1; N W W H 3087

# MODULE INFORMATION:

2170	200	9		
H6088	<b>6614H</b>	н9999		
	H	11		
CODE AREA SIZE	VARIABLE AREA SIZE =	MAXIMUM STACK SIZE	356 LINES READ	0 PROGRAM ERROR(S)

END OF PL/M-88 COMPILATION

# PL/M-80 COMPILER

ISIS-II PL/M-80 V3.0 COMPILATION OF MODULE SCPUBZ NO OBJECT MODULE REQUESTED COMPILER INVOKED BY: PLM80 :F1:SCPUB2.SRC NOOBJEC

PLM80 :F1:SCPUB2. SRC NOOBJECT PAGEWIDTH(80) PAGELENGTH(35)

ğ SCPUB2:

# NOLIST

PROCEDURE (P1, P2, P3) EXTERNAL; SYSMON: 400 6 8

(P2, P3) BYTE, P1 ADDRESS, ಶ್ವ

END SYSMON:

SETPERTIME

SET NEXT CALL TIME FOR PERIODIC

INPUT: P1 - PERIODIC INDEX

SETPERTIME: PROCEDURE (P1) PUBLIC: 3

(Pt. I. J. K. CY) BYTE. ಶ 52

XA1 = PERLIST(P1), PERINTADR.

CY = 8;

NN

2 2

/\* CLEAR CARRY \*/

N

```
IF CARRY THEN CY = 13
    DO J = 0 TO LAST(RTC);
          K = CY + RTC(1)
NNMMMM
5825894
```

ELSE CY = 0; K = K + PERINT(1);

/\* COMPUTE NEXT CALL TIME \*/ IF CARRY THEN CY = 1;

PERLIST(P1), PERTIME(I) = K

I = I - 1;

END;

MMMMNN

248868

END SETPERTIME; RETURN

安在安安林安安长大安全共和安全大安全大安全大学中华安全大学中华安全中华安全市中华安全市中华安全市中华安全市中华市中华市中华 \*\*

TIMECHE

: ADDRESS OF TIME (RTC FORMAT) CHECK RTC AGRINST INPUT TIME INPUT

TRUE IF INPUT TIME < RTC CUTPUT

FRLSE OTHERWISE

PROCEDURE (P1) BYTE PUBLIC;

TIMECHK:

4

69

(TIME1 BASED P1) (4) BYTE; P1 RDDRESS, I BYTE, 50 N 28

DO I = 0 TO LAST(RTC); N

M

# PL/N-80 COMPILER

IF TIME1(I) < RTC(I) THEN RETURN TRUE; IF TIME1(I) > RTC(I) THEN RETURN FALSE; END; RETURN FALSE; END TIMECHK;	**************************************	ACTIVATE PERIODIC CALL OF A PROCEDURE  INPUT : P1 - ADDRESS OF PROCEDURE  P2 - ADDRESS OF TIME INTERVAL(RTC FORMAT)	** ***********************************	PERACT: PROCEDURE (P1, P2) BYTE PUBLIC;	DCL (P1, P2) ADDRESS;	DO XB1 = 0 TO LAST(PERLIST); /* SEARCH FOR FREE SLOT */	IF PERLIST(XB1). FREE THEN GO TO PERACT1; /* FREE SLOT FOUND */	END;	CALL SYSMON(STACKPTR, 0, 3);		PERACT1: PERLIST(XB1), FREE = FALSE;	PERLIST(XB1), PERROR = P1;	PERLIST(XB1), PERINTADR = P2;
M M M N N				त	N	N	M	M	N	N	N	N	7
54878				62	88	81	85	20	83	86	87	80	89

DO XB2 = 0 TO LAST(RTC); /* SET NEXT CALL TIME */	PERLIST(XB1), PERTIME(XB2) = RTC(XB2), END:	PERXIBL(NUMPER) = XB1.	RUMPEK = NUMPEK + 1; RETURN X81;	END PERACT;	*	**************************************	***	CHANGE THE TIME INTERVAL OF A PERIODIC CALL	INPUT: P1 - PERIODIC INDEX	P2 - ADDRESS OF NEW TIME INTERVAL	**	安安女子并不安安女子女子女子女子女子女子女子女子女子女子女子女子女子女子女子女子女子女子女	PERCHG: PROCEDURE (P1, P2) PUBLIC:	DCL P1 BYTE, P2 ADDRESS;	IF NOT LEGAL(P1, MAXPER, 4) THEN RETURN.  /* ILLEGAL PERIODIC INDEX */	PERLIST(P1) PERINTADR = P2, /* SET NEW TIME INTERVAL */	CALL SETPERTIME(P1), /* SET NEW CALL TIME */	RETURN	END PERCHS
N	MM	N	NN	7									7	N	N	N	N	N	V
8	28	88	K 88	36									26	8	8	161	162	163	104

n

### PL/M-80 COMPILER

*
法非法未非法法法法法法法法法法法法法法法法法法法法法法法法法法法法法法法法法法
法非法法法法法法法法法法法法法法法法法法法法法法法法法法法法法法法法法法法法法
**

PERSUSP

SUSPEND THE PERIODIC CALL OF A PROCEDURE INPUT: PERIODIC INDEX

PROCEDURE (P1) PUBLIC: PERSUSP: 165

P1 BYTE;

SCL

186

IF NOT LEGAL (P1, MAXPER, 4) THEN RETURN;

/\* ILLEGAL PERIODIC INDEX \*/
PERLIST(P1), FREE = TRUE,

IF NUMPER = 0 THEN GO TO SUSP1; NUMPER = NUMPER - 13

NNNNN

109 110 111 113 114

167

XB2 = FALSE;

DO XB1 = 0 TO NUMPER;

/\* REMOVE P1 FROM PERXTBL AND COMPACT PERXTBL \*/ IF NOT XB2 AND PERXTBL(XB1) = P1 THEN XB2 = TRUE;

IF XB2 THEN PERXTBL(XB1-1) = PERXTBL(XB1);

END;

115

PERXTBL (NUMPER) = 0FFH; SUSP1:

RETURN

END; END SCPUB2; HNNNNMM

# PL/M-89 COMPILER

ISIS-II PL/M-80 V3.0 COMPILATION OF MODULE SCPUB3 NO OBJECT MODULE REQUESTED

Р <b>АGELENGTH</b> (35)				******
DOMPILER INVOKED BY: PLM88 :F1:SCPUB3. SRC NOOBJECT PRGENIDTH(80) PRGELENGTH(35)			بَ	/* ***********************************
MOOBJECT		SYSMON: PROCEDURE (P1, P2, P3) EXTERNAL; DCL (P2, P3) BYTE, P1 ADDRESS; END SYSMON:	PERACT: PROCEDURE (P1,P2) BYTE EXTERNAL) DCL (P1,P2) RDDRESS; END PERACT;	**********
98		P3) RDDRE	C BYT	****
CPUE3	N	P1, P2 E, P1	P1, P2 RESS.	****
F1:5	UPDATE: 2	JRE ( BYT SMORE	JRE ( RDD RRCT)	***
PLM86 :	7	PROCEDURE (P1, P2, P3) EXT (P2, P3) EXTED P1 RODRESS: END SYSMON.	PROCEDURE (P1, P2 (P1, P2 (P1, P2) RDDRESS; END PERACT)	*****
ED BY:	*/ \$NOLIST	SMORE	SACT:	***
COMPILER INVOKED BY: PLNSC SCPUB3: E	***	SYSI DCL	PER	* * *
T A S		400	400	
COMPI		4 4 N	222	

HANDLE MSG OVERFLOW

\*\*

MSGOVERFLOW

MSGOVERFLOW: PROCEDURE,

4

IF EXPERX <> OFFH THEN RETURN;  BU = MSG(SMN);  CODESAVE = R4;  EXPERX = PERACT(EXPERADR, 0);  RETURN;  END MSGOVERFLOW;	A*************************************	## CNOT A PUBLIC PROCEDURE:  INPUT: ADDRESS OF MCB  ***********************************	A ACC
~~~~~			1 0
85888888888888888888888888888888888888		G	3 4

MSGBUFFER(MSGIN) = MCB(B);

MSGIN = MSGIN + 1.

RETURN: END PACKMOB;

END;

NNMMNN

386888

DO B = 0 TO LAST (MCB),

SEND

\* RECEIVING MODULE NUMBER SEND A MESSAGE TO ANOTHER NODULE IN THE SYSTEM : ADDRESS OF MSG CONTROL BLOCK (MCB) MCB . RMN コミアして

\* SENDING MODULE NUMBER ME

\* MSG NUMBER Ξ

\* MSG LENGTH F

: FALSE IF RECEIVING MODULE IS NOT ACTIVATED OR DOES NOT EXIST OR RMW IS ILLEGAL CUTFUT

TRUE OTHERWISE

ADRMSGDATA IS SET TO ADDRESS OF FIRST DATA BYTE

PROCEDURE (P1) BYTE PUBLIC: SEND

(P1) ADDRESS, SAVEMSGIN ADDRESS, SC N

72

71

(MSG BASED P1) (4) BYTE,

IF NOT BIT(1,MODSTATUS(MSG(RMN))) OR MSG(RMN) > MAXSYSMOD N

THEN RETURN FALSES

- 1

2

/\* RECEIVING MODULE NOT ACTIVATED OR DOES NOT EXIST \*/ IF MSGOUT = MSGIN AND NUMMSG <> 0 THEN GO TO SENDS; /\* MSG BUFFER OVERFLOW \*/

N

35

IF MSGOUT <= MSGIN THEN

NNM

782

IF (LAST(MSGBUFFER) - MSGIN) >= (MSG(ML) - 1) THEN

GO TO SEND1;  /* MSG FITS INTO REST OF BUFFER */ LASTMSG = MSGIN; MSGIN = 0; END; IF (MSGOUT - MSGIN) < MSG(ML) THEN GO TO SEND2; /* NOT ENOUGH SPACE FOR MSG, OVERFLOW */	SEND1: SAVEMSGIN; CALL PACKMCB(P1); /* TRANSFER MCB */ MSGIN = SAVEMSGIN + MSG(ML); /* COMPUTE BEGIN FOR NEXT MSG */	IF MSGIN > LAST(MSGBUFFER) THEN  /* MSGIN IS OUTSIDE MSGBUFFER */  DO;  LASTMSG = MSGIN;  MSGIN = 0;	END; NUMMSG = NUMMSG + 1; ADRMSGDATA = .MSGBUFFER(SAVEMSGIN + 4); /* SET ADDRESS OF FIRST DATA BYTE */ RETURN TRUE; GFND2: ADRMSG = P1;	CALL RETU END	**************************************
м ммма	00 0	0 0 0 0 0		1000	
8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8	8 8 8	8 8 4 8	228 83	98 98 168	

	×
L	רהא
2	֓֞֝֝֝֞֝֝֞֝֝֟֝֝֟֝֓֞֝֞֩֓֓֓֞֩֞֩
Č	3
9	S
4	Ę
	J

WAIT FOR EXTMSGLOCK TO BECOME UNLOCKED AND LOCK	**************************************	PROCEDURE;	DO WHILE EXTMSGLOCK <> 0,  * EXT MSG BUFFER LOCKED */		EXTMSGLOCK = CPTR; /* SET LOCK */	IF EXTMSGLOCK <> CPTR THEN GOTO SETL1; /* ALREADY LOCKED BY OTHER COMPUTER */		TLOCK;		法全部法律法律法律法律法律法律法律法律法律法律法律法律法律法律法律法律法律法律法	UNLOCK	UNLOCK EXTERNAL MSG BUFFER		*************************************		PROCEDURE;	.0CK = 0;	***************************************	
FOR EXTMSG	*********		DO WHILL	END;	EXTMSGLOCK	IF EXTMSGLO	RETURNS	END SETLOCKS		***********		OCK EXTERNAL		*****		PROCEDU	EXTMSGLOCK = 0; RETURN; FND (IN) OCK:	*********	
**	*******	SETLOCK:	SETL1:						*/	******		UNIC.	**	******	*	ON LOCK:		*******	
		<del></del>	C)	M	OJ.	61	0	c)								ત	01010	ı	
		161	102	103	164	165	167	168								103	110		

SETEXTMSG

COMPUTE NUMBER OF RECEIVING COMPUTER AND SET VARIABLE EXTMSG

SETEXTMSG: PROCEDURE, 113

(X1, X2, X3) BYTE; SC

THEN EXTMSG = 0; IF NUMEXTMSG = 8

ELSE DO;

X1 = 8;X2 = EXIMSGBUFFER(EXIMSGOUT);

/\* KMN \*/

IF X2 < X1 THEN GOTO SET1. DO X3 = 1 TO MAXCPTR

END;

X1 = X1 + 8;

EXTMSG = X3;

SET1:

EMD; RETURN

END SETEXTMSG \* 

 RECEXT

115

117 118 119

^

# PL/M-80 COMPILER

TRANSFER MSG FROM EXTERNAL MSG BUFFER INTO OWN

82 12 12 12 12 12 12 14 14 14 14 14 14 14 14 14 14 14 14 14
-------------------------------------------------------------

NUMEXTMSG = NUMEXTMSG - 15

REC1:

N

œ

CALL SETEXTMSG;  /* SET NUMBER OF RECEIVING CPTR OF NEXT MSG */	CALL UNLOCKS	END RECEXT,	/* ***********************************	**************************************	**	SENDEXT	SEND MSG TO EXTERNAL MODULE	OUTPUT: TRUE IF MSG SENT	FALSE OTHERWISE	**	我是不够是不会是不会是不是不是不是不是,我们的,我们也是我们的,我们也是不是不是不是不是不是不是不是不是不是不是	SENDEXT: PROCEDURE BYTE PUBLIC;	CALL SETLOCK	/* SET EXT MSG BUFFER LOCK */	IF EXTMSGIN = EXTMSGOUT AND NUMEXTMSG <> 0 THEN GOTO SEXT2	* MOJERT ( * )	IF EXIMSGOUT <= EXIMSGIN THEN	DQ;	IF LAST(EXTASGBUFFER) - EXTASGIN >= MSG(ML) THEN GOTO SE	- XT1:	/* MSG FITS INTO REST OF BUFFER */	EXTLASTMSG = EXTMSGIN	EXTMSGIN = 8;
N	NO	וייי										7	N		0	•	N	N	M			M	M
148	149	151										152	153		154		126	157	158			169	161

a

END; IF EXTMSGOUT - EXTMSGIN < MSG(ML) THEN GOTO SEXTZ; /* OVERFLOW */	SEXT1: XB2 = MSG(ML) - 1; D0 XB3 = 0 T0 XB2; /* TRANSFER MSG */	EXIMSGIN = EXTMSGIN + 1; EXTMSGIN = EXTMSGIN + 1; END;	IF EXTMSGIN > LAST(EXTMSGBUFFER) THEN  /* EXTMSGIN OUTSIDE BUFFER */	DO; EXT GOING = EXTMEGIN:	EXTMSGIN = 6:	El-D;	NUMEXTMSG = NUMEXTMSG + 1;	IF NUMEXTMSG = 1 THEN CALL SETEXTMSG; /* SET RECEIVING CPTR OF NEXT MSG */	CALL UNLOCK;  /* UNLOCK EXT MSG BUFFER */	RETURN TRUE;	SEXT2: CALL MSGOVERFLOW:	CALL UNLOCK;	RETURN FALSE;	END SENDEXT;	*	安全不安全不会不会不会不会不会不不会不不会不不会不会不会不会不会不会不会不会不不会不不不不	*************************************	77
MN	00 m	M M M	N	0 r	n M	M	N	N	N	N	N	8	8	8				
162 163	165	168 168 169	178	171	173	174	175	176	178	173	180	181	182	183				

ILLEGRLMSG

#### PL/M-80 COMPILER

PROCESS ILLEGAL MSG RECEIVED BY A MODULE

	410
	*
	×
	7.
	*
	*
	7.
	*
	*
	*
	*
	-
	~
	-
	*
	*
	•
	::
	•
	:
	+
	3
	+
	3
	+
	*
	::
	•
	•
	•
	::
	•
	+
	:
	•
	*
	*
	*
	*
	::
	*
	*
	*
	*
	***
	***
	***
	****
	****
	****
	*****
	*****
	*****
	*****
	*****
	*****
	******
	*****
	******
	*****
	******
	******
	******
	*****
	*****
	*****
	******
	*****
	*****
	******
	******
	******
	******
	*******
	*******
	*******
	******
	*******
	*******
	*********
	*********
	********
	*********
	**********
	***********
	*********
	***********
	***********
	**********
	**********
	**********
•	************
	***********
	***********
	**********
	************
	**********
	**********

	707
	2
	2
	2
PUBLIC	0707
ZKE.	DUTE
ILLEGALMSG: PROCEDURE PUBLIC	Security and any other new 48 48
HLMSG:	200
ILLEG	2
ત	c
184	100

PRSYNC(4) BYTE DATA (CS, EX, 10, 40);	XB2 = 16;	DO XB1 = 0 TO LAST(MSG);	CALL CONVASC(MSG(XB1))	ILLMSG(XB2) = BU;	ILLMSG(XB2 + 1) = BL	XB2 = XB2 + 6;	END;	/* CONVERT RMN SMN, MN, ML */	IF NOT SEND(, PRSYNC(0)) THEN RETURN:	/* SEND SYNC PRINT MSG TO CRT */	DO XB1 = 0 TO LAST(ILLMSG);	MSGDRTR(XB1) = ILLMSG(XB1);	END;	RETURN	END ILLEGALMSG;		
700																END;	
N	N	N	M	M	M	M	M		Ø		N	M	M	N	N	7	
185	186	187	188	189	196	191	192		193		195	196	197	198	199	288	

# MODULE INFORMATION:

9450	130	9
63В1Н	HOBBB	ВВВЕН
11	11	ti
	SIZE	SIZE
SIZE	FREA	THCK
AREA	RELE	MUM S
CODE	VARIABLE	MAXI

# PL/M-80 COMPILER

PLM80 :F1:SCPUB4, SRC NOOBJECT PRGEWIDTH(80) PRGELENGTH(35) ISIS-II PL/M-80 V3.0 COMPILATION OF MODULE SCPUBA NO OBJECT MODULE REQUESTED COMPILER INVOKED BY: PLM80 :F1:SCPUB4 SRC MOOBIF

		EXTERNAL;
	, 00,	PROCEDURE (P1, P2, P3) EXTERNAL, (P2, P3) BYTE, P1 ADDRESS; END SYSMON;
	SCPUB4:	SYSMON: DCL
	4	58 4 88 58 9 2 4 2 4 2 4 2 4 4 5 4 5 4 5 4 5 4 5 4 5
)	,	-

PRIORITYINT

(PRIORITYINT TO BE CALLED FROM INT ROUTINES ONLY) SCHEDULE THE CALL OF A PRIORITY PROCEDURE INPUT: PRIORITY INDEX PRIORITYINT: PROCEDURE (P1) PUBLIC:

27

IF P1 >= MAXPRIOR THEN RETURN P1 BYTE; ರ್ಷ N N 52 5

/\* ILLEGAL PRIORITY \*/
IF P1 (> 0 N 33

1LEF	
ш	
_	
_	
$\overline{}$	
u_	
=	
50	
=	
-	
$\mathbf{\omega}$	
_	
-	
123	
8	
m	
-	
-	
ξ	
_	
1	
•	
200	
7	
n	
-	

THEN $P1 = ROL(1, P1);$	ELSE P1 = 1;	PRIORSCHEDULE * PRIORSCHEDULE OR P1;	RETURN	END PRIORITYINT;	*/	安全各种的人名英格兰 医克里氏病 计分别 医克里氏病 医克里氏病 医多种性 医多种性 医多种性 医多种性 医多种性 医多种性 医多种性 医多种性	安全的安全的安全的安全的安全的安全的安全的安全的安全的安全的安全的安全的安全的安	**	PRIORITY	
	N	8	N	N						
	52	28	29	68						

SCHEDULE THE CALL OF A PRIORITY PROCEDURE

INPUT: PRIORITY INDEX

PROCEDURE (P1) PUBLIC: PRIORITY: 61

P1 BYTE; ಶ್ವ N 62

IF P1 >= MAXPRIOR THEN RETURN N 63

CALL SETBIT(P1. PRIORSCHEDULE); /\* SET SCHEDULE BIT \*/ RETURN N

65

/\* ATTEMPT TO SCHEDULE ILLEGAL PRIORITY \*/

NN 96

END PRIORITY:

\*\*

ENTERPRIOR

M

OUTPUT : PRIORITY INDEX TO PRIORLIST

ENTERPRIOR: PROCEDURE (P1) BYTE PUBLIC: 68

DO XB1 = 0 TO LAST (PRIORLIST); P1 HDDRESS; ಶ್ವ 69

/\* SEARCH FOR FREE SLOT IN PRIORLIST \*/

IF PRIORLIST(XB1) = 0 THEN

/\* ADDRESS OF PRIORITY PROCEDURE \*/ PRIORLIST(XB1) = P1;

RETURN XB1;

EMD;

2222

/\* PRIORITY LIST OVERFLOW \*/ CALL SYSMON(STRCKPTR, 8, 2); END;

RETURN BFFH;

END ENTERPRIOR:

NN

36

REMPRIOR

REMOVE PRIORITY CALL FROM PRIORLIST INPUT: PRIORITY INDEX

\*\*

\*\*

222

**************************************	REMPRIOR: PROCEDURE (P1) PUBLIC:	DCL P1 BYTE:	IF NOT LEGAL(P1,MAXPRIOR,6) THEN RETURN; /* ATTEMPT TO REMOVE PRIORITY WITH ILLEGAL INDEX */	CALL CLEARBIT(P1, PRIORSCHEDULE); /* CLEAR SCHEDULE BIT IF SET */	PRIORLIST(P1) = 0; /* REMOVE PRIORITY ADDRESS */	RETURN	END REMPRIOR:	*/	**************************************	***************************************	SETCDC	SET CDC INTERRUPT  INPUT : P1 - TIME INTERVAL (LSB = 1.86 MICRO SEC P2 - ADDRESS TO BE CALLED AT CDC INT OUTPUT: FALSE IF CDC ALREADY ACTIVE TRUE OTHERNISE	**	*************************************	SETCDC: PROCEDURE (P1, P2) BYTE PUBLIC:	DCL (P1, P2) ADDRESS, (P1H, P1L) BYTE AT (. P1);
	7	N	N	N	N	7	N								4	N
	88	87	82	<u>%</u>	8	8	82								88	89

n

IF CDCACTIVE THEN RETURN FALSE; CDCACTIVE = TRUE; DISABLE;	INTMASK * INTMASK XOR 1;	OUTPUT(608H) = INTMASK;	ENABLE	OUTPUT(60FH) = 01110060B;	/* CTR 1 IN MODE 8 */	OUTPUT(600H) = P1H;	OUTPUT(600H) = P1L;	/* LOAD CTR 1 */	CDCRDR = P2;	/* SAVE ADRRESS TO BE CALLED AT CDC INT */	RETURN TRUE;	END SETCDC:	*	**************************************	**************************************	**	ENTERINT
000	0	N	N	N		N	N		N		N	N					
888	24	95	36	26		88	88		186		161	162					

ENTER INTERRUPT
INPUT : P1 - INT LEVEL
P2 - PRIORITY INDEX
OUTPUT: FALSE IF INT ALREADY SET
TRUE OTHERWISE
\*\*\*

#/
103 1 ENTERINT: PROCEDURE (P1, P2) BYTE PUBLIC:

×
1LER
COMP
_
8
¥
7
5

<pre>/* ENABLE INT LEVEL P1 */ ENABLE; RETURN TRUE; END ENTERINT; /* **********************************</pre>	000 400	41 31 71 81 19
ENFBLE; RETURN TRUE; END ENTERINT;	<b>01</b> 01 0	115
DISABLE; CUTPUT(8DBH) = INTMASK; /* ENABLE INT LEVEL P1 */	NN	112
<pre>INTIBL(P1) = P2; /* SET PRIORITY INDEX FOR INT LEVEL */ INTMASK = INTMASK XOR SHFT(P1);</pre>	N N	1116
DO: CALL SYSMON(STACKPTR, 0,7); RETURN FALSE;	<b>0</b> M M	186 187 188
IF INTTBL(P1) <> OFFH THEN  /* INT LEVEL ALREADY SET */	N	165
DCL (P1, P2) BVTE,	N	164

### PL/M-89 COMPILER

INTMASK = INTMASK OR SHFT(P1);	DISABLE	OUTPUT(00BH) = INTMASK;	ENABLE	RETURN	END REMINT;	END SCPUB4;
N	N	N	N	N	N	4
128	121	122	123	124	125	126

### MODULE INFORMATION:

371D	12D	4		
= 6173H	= 666CH	<b>6664H</b>		
11	u	11		
CODE AREA SIZE	VARIABLE AREA SIZE	MAXIMUM STACK SIZE = 0004H	434 LINES READ	CONGROUND MAGELIAN R

# END OF PL/M-80 COMPILATION

ISIS-II PL/M-80 V3. 0 COMPILATION OF MODULE SCPUBS

NO OBJECT MODULE REQUESTED COMPILER INVOKED BY: PLASSPUBS. SRC NOOBJECT PROSEMIDTH(80) PROSELENGTH(35)

PROCEDURE (P1) ADDRESS EXTERNAL; P1 ADDRESS; END SEND; ĝ SCPUBS #NOL1ST SEND S HNN 58 48

ACTIVATE

ACTIVATE A SUSPENDED MODULE

: P1 - NUMBER OF MODULE TO BE ACTIVATED P2 - NUMBER OF CALLING MODULE INPUT

: FALSE IF CALLING MODULE NOT AUTHORIZED OR IF MODULE DOES NOT EXIST

TRUE IF O. K.

CUTFUT

我是我看着我们是我们的,我们是我们的,我们们的,我们们的,我们们们的,我们们们的,我们们们的一个,我们们们们们们们们们们的。

PROCEDURE (P1, P2) BYTE PUBLIC

(P1, P2) BYTE; DC D N 52

ACTIVATE:

51

53

Ø IF NOT BIT(3, MODSTATUS(P2)) OR MODSTATUS(P1) = N

THEN RETURN FALSE;  /* CALLING MODULE NOT AUTHORIZED OR MODULE DOES  NOT EXIST */	CALL SETBIT(1, MODSTATUS(P1)); /* SET ACTIVE BIT */	IF NOT SEND(.SYSNN1(0)) THEN RETURN FALSE; /* SEND RESTART MSG TO MODULE */	RETURN TRUE; END ACTIVATE;	/* ***********************************	***	SUSPEND A CURRENTLY ACTIVE MODULE  INPUT : P1 - NUMBER OF MODULE TO BE SUSPENDED  P2 - NUMBER OF CALLING MODULE	OUTPUT : FALSE IF CALLING MODULE NOT AUTHORIZED OR IF MODULE CANNOT BE SUSPENDED	TRUE IF O. K.	** ***********************************	SUSPEND: PROCEDURE (P1, P2) BYTE PUBLIC;	DCL (P1, P2) BYTE;	IF NOT BIT(3,MODSTATUS(P2)) OR NOT BIT(2,MODSTATUS(P1)) THEN RETURN FALSE;  /* CALLING MODULE NOT AUTHORIZED OR MODULE CANNOT BE SUSPENDED */
	Ø	Ø	0101							त	N	N
	B	8	88							68	61	62

M

c	¥	•
ī	i	į
	_	Ì
•		
ú	ì	
ě	į	
		ļ
ĺ	-	
	,	
C	ì	į
ļ	Ņ	
	ı	

IF NOT SEND(. SYSMN5(0)) THEN RETURN FALSE;	/* SEND SUSPEND MSG */	CALL CLEARBIT(1, MODSTATUS(P1));	/* CLEAR ACTIVE BIT */	RETURN TRUE;	END SUSPEND;	*	老者我们是我们的人们的人们的人们的人们的人们的人们的人们的人们的人们的人们的人们的人们的人们	安安安安安安安安安安安安安安安安安安安安安安安安安安安安安安安安安安安安安安安	**	ACT I VE
N		N		8	N					
49		99		29	68					

CHECK ACTIVE STATUS OF MODULE INPUT : MODULE NUMBER OUTPUT : TRUE IF MODULE IS ACTIVATED, FALSE IF NOT

PROCEDURE (P1) BYTE PUBLIC:

DQ DC N

ACTIVE:

69

P1 BYTE; 92

RETURN BIT(1,MODSTATUS(P1)); END ACTIVE; END SCPUBS; 204 222

### MODULE INFORMATION:

1810 50 = 88B5H VARIABLE AREA SIZE = 0005H CODE AREA SIZE

ISIS-II PL/M-80 V3. 0 COMPILATION OF MODULE SCPUBG

NO OBJECT MODULE REQUESTED

COMPILER INVOKED BY: PLM80 :F1:SCPUB6.5RC NOOBJECT PRGENIDTH(80) PRGELENGTH(35)

ĝ SCPUB6: \$NOLIST 4

\*\*

UPDSTRT

UPDATE MODULE STATUS IN MODSTATUS TABLE

INPUT: P1 - MODULE NUMBER

P2 - STATUS

PROCEDURE (P1, P2) PUBLIC: UPDSTRT

38

(P1, P2) BYTE; SC N 31

MODSTRTUS(P1) = P2NNN

END UPDSTRT; 222

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 

PROCADR

N

### PL/M-80 COMPILER

```
RETURN ADDRESS OF BEGIN OF CALLING PROCEDURE
```

PROCEDURE ADDRESS PUBLICA PROCADR: 33 (A1, A2) ADDRESS, (A2U, A2L) BYTE AT (. A2), (STACKTOP BASED A1) (2) BYTE) ಶ್ವ N

36

R2U = STACKTOP(6) - 11; R1 = STACKPTR;

RZL = STRCKTOP(1) MINUS 0;

END PROCADR: RETURN AZ;

CONVERT 2 DIGIT HEX NUMBER INTO 2 ASCII CODES

CONVASC

: NUMBER (BYTE) INPUT

OUTPUT : RSCII CODES IN R4

PROCEDURE (P1) PUBLIC: CONVASC 42

P1 BYTE; ಶ್ವ N 43

BL = (P1 FND 68681111B) + 38H;

N

44

NNNNN

W 8 8 4 4

M

IF BL > 39H THEN BL = BL + 7;	BU = ROR((P1 AND 11116866B), 4) + 38H;	IF BU > 39H THEN BU = BU + 7;	RETURN	END CONVASC;	
8	2	2	2	2	₹

54488

DEBLK

MOVE POINTER B1 TO 1ST NOM-BLANK IN MSGDATA STARTING AT MSGDATA(B1)

DO WHILE MSGDATA(B1) =  $^{\prime}$  AND B1 <= MSG(ML) -  $^{\prime}$ 5 PROCEDURE PUBLIC: B1 = B1 + 1RETURN END; DEBLK NMMNN 24886 52

END DEBLKS

\*\*

GETNUM

CONVERT ASCII CODES IN MSGDATA TO HEX NUMBER CONVERTED NUMBER IS LEFT IN 84 STARTIN AT MSGDATA(B1)

# RETURN TRUE IF NUMBER IS O. K. , FALSE OTHERNISE

		**************************************
88	ч	*/ GETNUM: PROCEDURE BYTE PUBLIC;
59	8	F4 = 6;
69	2	XB3 = MSG(ML) - 5;
61	8	IF 81 > XB3 THEN RETURN TRUE;
63	2	XB2 = B1 + 3;
64	2	DO B1 = B1 T0 XB3;
		/* CONVERT TO END OF MSGDATA, BUT MAX. 4 DIGITS */
65	М	XB1 = MSGDRTR(B1);
99	M	IF XB1 = ' ' OR XB1 = ',' OR XB1 = '-' THEN RETURN TRUE
		/* / / RND /, RND /-/ RRE LEGRL DELIMITER */
68	M	IF B1 > XB2 THEN RETURN FALSE;
		/* NUMBER TOO LARGE */
92	M	R4 = SHL(R4, 4);
71	M	IF XB1 >= '8' FND XB1 <= '9'
		THEN XB1 = XB1 - '0';
23	м	ELSE IF XB1 < 'A' OR XB1 > 'F'
		THEN RETURN FALSE;
		/* ILLEGAL CHARACTER */
75	M	ELSE XB1 = XB1 - 37H;
32	M	R4 = R4 + XB1;
22	M	END;
28	2	RETURN TRUE;
23	2	END GETNUM;
86	1	END SCPUB6;

### PL/M-80 COMPILER

ISIS-II PL/M-80 V3.0 COMPILATION OF MODULE SCPUB? NO OBJECT MODULE REQUESTED COMPILER INVOKED BY: PLN80 :F1:SCPUB? SRC NOOBJECT PAGEWIDTH(80) PAGELENGTH(35)

ô SCPUB7: \$NOLIST PROCEDURE (P1) EXTERNAL; P1 BYTE; CONVASC: DECLARE 400400 

END

PROCEDURE (P1) BYTE EXTERNAL, SEND:

P1 RDDRESS; END SEND; DECLARE

SYSMON

SYSTEM MONITOR

MAY BE CALLED WHEN AN INTERNAL ERROR CONDITION OCCURS, SYSMON GENERATES AN ASYNCHRONOUS PRINTOUT TO INDICATE PROGRAM LIMITS ARE EXCEEDED ETC.

INPUT: P1 - LOCATION OF ERROR OR OTHER ADDRESS VALUE THE NATURE AND LOCATION OF ERROR CONDITION.

P2, P3 - BYTE VALUES TO SPECIFY ERROR

### PL/M-80 COMPILER

			2	3				
			11	11				
PROCEDURE (P1, P2, P3) PUBLIC:	P1 ADDRESS, (P2, P3, I) BYTE, (STACKTOP BASED P1) (2) BYTE;	CALL CONVASC(STACKTOP(1)); MONTEXT(16) = BU, MONTEXT(17) = BL; CONT. CONVASC(STACKTOP(2));	COLL COMMISCASINGNIONOS. MONTEXI(18) = BU; MONTEXI(19) = BL; CBL: COMMISCAPS; MONTEXI(21) = RL; MONTEXI(22) = RL;	CALL CONVASC(P3), MONTEXT(24) = BU, MONTEXT(25) = 15 NOT SEND( MONMSG(A)) THEN PETIEN:	/* SEND ASYNC PRINT MSG TO CRT */ DO I = 0 TO LAST(MONTEXT)	MSGDATA(I) = MONTEXT(I); END;	RETURN; END SYSMON;	1875
SYSNON:	DECLARE							END SCPUBZ:
SYS	DEC							E P
4	01	000	100	100	1 0	мм	0101	<b>+</b>
36	32	868	4 4 4	54 2	3 33	2 t	8 8 8	22

### MODULE INFORMATION:

CODE AREA SIZE	11	= 6692H	146D	
VARIABLE AREA SIZE	11	= 6665H	20	
37 IZE	11	= 6662H	20	
265 LINES READ				
8 PROGRAM ERROR(S)				

# END OF PL/M-88 COMPILATION

DEBUG MODULE

4

### PL/M-88 COMPILER

ISIS-II PL/M-80 V3.0 COMPILATION OF MODULE DBMOD1 NO OBJECT MODULE REQUESTED COMPILER INVOKED BY: PLM80 :F1:D81.5RC NOOBJECT PAGEWIDTH(80) PAGELENGTH(35)

# INCLUE(:F1:DEDATP.SRC)  ***********************************			
		DBM0D1: D0:	TP SPC
	"		
	11	******	安安安安安安安安安安安安安安安安安安安安安安安安安安安安安安安安安安安安安安安
	11	******	*************************************
	11	**	
	#	**	DEBUG MODULE
	II	**	
	H	********	**************************************
	H	**	
	II	OW **	ULE IDENTIFICATION : DB
	11	ION **	ULE NUMBER : 67
	11	**	
	11	*******	**************************************
	11	*/	
	11	# NOLIST	
	11	*	
	11	*********	**************************************
	11	**	
	11	**	DB DATA
	11	**	
= ** = ** = */ = DCL PROMPT LIT '3EH',	11	*******	**************************************
= ** SUBSTITUTIONS = */ = DCL PROMPT LIT '3EH',	11	**	
= */ = DCL PROMPT LIT '3EH',	11	**	SUBSTITUTIONS
= DCL PROMPT LIT '3EH',	11	14	
	1 =	DCL PROMPT LIT /3	

N

```
*******************************
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      DEBUG REQUEST MSG TO EXT. DEBUG MODULE
                                                                                                                                                                                          (4) BYTE PUBLIC INITIAL (0, DB, 10, 0),
                                                                                                                                                                                                                                 (4) BYTE PUBLIC INITIAL (0, DB, 11, 0),
                                                                                                                                                                                                                                                                    (4) BYTE PUBLIC INITIAL (0, DB, 12, 5),
                                                                                                                                                                                                                                                                                                           (4) BYTE PUBLIC INITIAL (0, DB, 14, 4),
                                                                                                                                                                                                                                                                                                                                                (4) BYTE PUBLIC INITIAL (0, DB, 15, 4),
                                                                                                                                                                                                                                                                                                                                                                                      (4) BYTE PUBLIC INITIAL (0, DB, 16, 4),
                                                                                                                                                                                                                                                                                                                                                                                                                              (8, DB, 17, 4),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                  (0, DB, 18, 0),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        (4) BYTE PUBLIC INITIAL (0, DB, 24, 5),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     SYNC PRINT MSG, TELLBACK REQUIRED */
                                                                                                                                                      MESSAGE CONTROL BLOCKS
                                                                                                                                                                                                                                                                                                                                                                   ACTIVATE DEBUG MSG TO CRT */
                                                                                                                                                                                                                                                                                                                                                                                                                                               INPUT LINE MSG */
                                                                                                                                                                                                                                                                                                                                                                                                                           (4) BYTE PUBLIC INITIAL
                                                                                                                                                                                                                                                                                                                                                                                                                                                                   (4) BYTE PUBLIC INITIAL
                                                                                                                                                                                                                                                                                                                                                                                                           NEW LINE MSG TO CRT */
                                                                                                                                                                                                                                                                                        INPUT REQUEST MSG */
                                                                                                                                                                                                                                                                                                                             TERMINATE 1/0 MSG */
                                                                                                                                                                                                             ASYNC PRINT MSG */
                                                                                                                                                                                                                                                  SYNC PRINT MSG */
                                                       FUNCTIONINPUT LIT 131,
                                    DEBUGCOMMAND LIT '2',
                SELECTOPTR LIT '1',
INITDEBUG LIT '8',
                                                                           ENDSUBST LIT '8';
                                                                                                                                                                                                                                                                                                                                                                                                                                                 BEGIN OF
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              ENDMCB LIT '8';
                                                                                                                                                                                          DEMN10
                                                                                                                                                                                                                                                                                                                                                                                                             *
                                                                                                                                                                                                                *
                                                                                                                                                                                                                                                      *
                                                                                                                                                                                                                                                                                                                                 *
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          *
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         DBMM24
                                                                                                                                                                                                                                                                                            *
                                                                                                                                                                                                                                                                                                            DEMM14
                                                                                                                                                                                                                                                                                                                                                 DBMM15
                                                                                                                                                                                                                                                                                                                                                                       *
                                                                                                                                                                                                                                                                                                                                                                                        DBMN16
                                                                                                                                                                                                                                                                                                                                                                                                                                                   *
                                                                                                                                                                                                                                                                                                                                                                                                                                                                   DEMM18
                                                                                                                                                                                                                                 DBMM11
                                                                                                                                                                                                                                                                      DBMM12
                                                                                                                                                                                                                                                                                                                                                                                                                              DEMM17
                                                                                                                                                                                         200
                                                                                                                                                      **
                                                                                                                                     **
```

\*\*

------

M

```
NOTACTMSG (26) BYTE PUBLIC INITIAL (0, ' DEBUG MODULE NOT ACTIV
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     MSGTEXTX (6) ADDRESS PUBLIC INITIAL (0,0602H,060AH,0512H,0519H
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    SKIPMSG (18) BYTE PUBLIC INITIAL (3,0,7 7,0,0,7MSG NOT TYPED/)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          FUNCTBL (10) BYTE PUBLIC INITIAL ('A', 'B', 'C', 'D', 'S', 'M', 'H',
                                                                                                                                                                                                                                                                                                       ERRMSG1 (22) BYTE PUBLIC INITIAL (1, CPTR ALREADY DEBUGGED'),
                                                                                                                                                                                                                                                                                                                                             TERMING (18) BYTE PUBLIC INITIAL (1, ' DEBUG TERMINATED'),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             SYSTTEXT1 (12) BYTE PUBLIC INITIAL (0, ' FUNCTION: '), SYSTTEXT2 (14) BYTE PUBLIC INITIAL (0, 'SUBFUNCTION: '),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    DATA
                                                                      SELCP1 (12) BYTE PUBLIC INITIAL (0, 'DEBUG CPTR:'),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  (4) BYTE PUBLIC INITIAL (EX, DB, 10, 4),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           EXTRACTSTOP (4) BYTE PUBLIC INITIAL (EX. DB, 11, 4),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      SYSTTEXT3 (7) BYTE PUBLIC INITIAL (8, ' DATA: '),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                    /* TIME INTERVAL FOR PERIODIC FUNCTIONS */
                                                                                                            SELCP2 (8) BYTE PUBLIC INITIAL (8, 72-CPTR: 7),
                                                                                                                                                     PROMPTMSG (2) BYTE PUBLIC INITIAL (0, PROMPT),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               SYSTMSG (4) BYTE PUBLIC INITIAL (3, DB, 50H, 0),
                                                                                                                                                                                                                                                                                                                                                                                                                              DEPERINT (4) BYTE PUBLIC INITIAL (6, 6, 6, 6),
                                                                                                                                                                                                                                                                      ERRMSG (3) BYTE PUBLIC INITIAL (8, 7 ?7),
                                                                                                                                                                                                                                                                                                                                                                                   INSPBUF (17) BYTE PUBLIC INITIAL (8,8,7
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        /* START EXTRACTION MSG TO EXEC */
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       7
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    /* STOP EXTRACTION MSG TO EXEC */
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   /* DEBUG COMMAND IDENTIFICATIONS
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       MSGTEXT (40) BYTE PUBLIC INITIAL
CONSTANT DATA
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            INDEX IS DEFUNCTION */
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    NAME
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  RIM
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  EXTRACTSTART
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               (1, 8, '
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                'Y', 8, 'T'),
                                                                         S
S
                                                                                                                                                                                                                                        É
```

```
/* MODULE NUMBER OF CRT MODULE CONNECTED TO DB */
                                                                                                                                                                                                                                                                 /* CRIINPUT AND DEFUNCTION CONTROL PROGRAM FLOW
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     /* TRUE IF DEBUG FUNCTION IN BYTE MODE */
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     /* TRUE IF TELLBACK MSG RECEIVED */
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             (DBINH, DBINL) BYTE PUBLIC AT (, DBIN),
                                                                                                                                                                                                                                                                                                                                                                                                        /* TRUE IF HARDCOPY REQUESTED */
                                                                                                                                                                                                             DEFUNCTION BYTE PUBLIC INITIAL (0),
                                                                                                       VARIABLE DATA
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        /* INPUT IN MACHINE FORMAT */
                                                                                                                                                                                                                                                                                              IN PROCEDURE DBINPUT */
                                                                                                                                                                                                                                                                                                                                                                                                                                                           /* CODE OF TELLBACK MSG */
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         DETELLERCK BYTE PUBLIC,
                                                                                                                                                          DCL DBCLEARBEG BYTE PUBLIC,
                                                                                                                                                                                                                                           CRIINPUT BYTE PUBLIC,
                                                                                                                                                                                                                                                                                                                                                                               HARDCOPY BYTE PUBLIC,
                                                                                                                                                                                                                                                                                                                                                                                                                                      DETECODE BYTE PUBLIC,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                BYTEMODE BYTE PUBLIC,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     DBIN ADDRESS PUBLIC,
                                                                                                                                                                                                                                                                                                                           DECRT BYTE PUBLIC,
ENDCONST LIT '8';
                                                                                                                                                              14
```

/\* START ADDRESS OF SNAPSHOT PROCEDURE \*/

/\* IF 8, FIRST CALL OF SNAPSHOT \*/

SNFL BYTE PUBLIC,

SNAPSHOTADR ADDRESS PUBLIC,

DEBUG BYTE PUBLIC,

EXTDE BYTE PUBLIC,

/\* TRUE IF CPTR ALREADY DEBUGGED \*/

n

```
/* TRUE IF TERMINATE PERIODIC FUNCTION MSG RECEIVED */
/* MODULE NUMBER OF EXT. DEBUG MODULE */
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     /* LAST BYTE TO TYPE IN A DUMP LINE */
                                                                                                                                                                   /* TRUE IF PERIODIC FUNCTION ACTIVE */
                                                                                                                                                                                                                                                                                                                                            (DBADRL, DBADRH) BYTE PUBLIC AT (. DBADR),
                                                    /* IF 8, FIRST CALL OF DBINPUT */
                                                                                                                                                                                                                                                                                                                                                                      /* ADDRESS OF DEBUG LOCATION */
                                                                                                         /* START ADDRESS OF DBINPUT */
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  /* SNAPSHOT PRIORITY INDEX */
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             /* INDEX OF DEBUG PERIODIC */
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               (DUMPBEG, DUMPEND) RDDRESS PUBLIC,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       (DUMPCONTA BASED A2) (1) ADDRESS, (DUMPCONTB BASED A2) (1) BYTE,
                                                                                                                                                                                                                                                                                                                CONTADR BASED DBADR ADDRESS,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          DEMSGDATA (20H) BYTE PUBLIC,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           /* TRUE IF DUMP DONE */
                                                                                                                                                                                                                                                                                                                                                                                                                                /* PREVIOUS FUNCTION */
                                                                                                                                                                                                                                                                                                                                                                                                                                                            DEGUTBUF (160) BYTE PUBLIC,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            DUMPBEGIMP ADDRESS PUBLIC,
                                                                               DBINPUTADE ADDRESS PUBLIC.
                                                                                                                                                                                                                                                                                    CONTRYTE BASED DBADR BYTE,
                                                                                                                                                                                               TERMPERFUNCT BYTE PUBLIC,
                                                                                                                                     PERFUNCTION BYTE PUBLIC.
                                                                                                                                                                                                                                                                                                                                                                                                    DBFUNCTSAVE BYTE PUBLIC,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        /* OUTPUT BUFFER */
                                                                                                                                                                                                                                                       DBADR ADDRESS PUBLIC,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 DUMPDONE BYTE PUBLIC,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           SNPRIORX BYTE PUBLIC,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           /* DUMP RANGE */
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         ENDLINE BYTE PUBLIC,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  DEPERX BYTE PUBLIC.
                          DBFL BYTE PUBLIC,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   11
                                                                                                                                                                                                                                                                                                                      11
```

```
OC3H, OC4H, OD4H, OE4H, OF4H, OCAH, ODAH, OERH, OFAH, 2RH, 3RH, OCCH
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    (003H, 6, 16H, 26H, 36H, 006H, 006H, 0E6H, 0F6H, 00BH, 0EH, 1EH, 2EH, 3
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                0CEH, 0DEH, 0EEH, 0FEH, 1, 11H, 21H, 31H, 0C2H, 0D2H, 0E2H, 0F2H, 22H
                                                                                                                                                                                                                                                                                                                                                                                                   /* NUMBER OF SAME MSG DURING THE PROCESS OF EXTRACTION */
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 /* BUFFER AND INDEX FOR SYSTEM TEST INPUTS */
                                                                                                                                                                                                                                                                                                                                       /* TRUE IF A MSG EXTRACTION IN PROCESS */
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              DESTBUF (20H) BYTE PUBLIC AT (. DBMSGDATA(0)),
                                                         /* CONTROLS INPUT OF 'M' COMMAND */
                                                                                                                                                                                                                                                                                                                                                                                                                                                                /* SWITCH FOR SYSTEM TEST INPUTS */
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      DCL INSTRIBL (44) BYTE PUBLIC INITIAL
                                                                                                                     /* TRUE IF MSG SIMULATION */
                                                                                                                                                                                /* TRUE IF MSG EXTRACTION */
/* DATA OF SINULATED MSG */
                                                                                                                                                                                                                                          /* INDEX TO DBMSGDATA */
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             SNAPSHOT DATA
                                                                                                                                                                                                                                                                                                                                                                     DEMSGCOUNT BYTE PUBLIC,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 DECLEAREND BYTE PUBLIC:
                             DBMSWITCH BYTE PUBLIC,
                                                                                                                                                                                                                                                                                                       DEEXTRFL BYTE PUBLIC,
                                                                                                                                                                                                                                                                                                                                                                                                                               PUBLIC,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            DESTRUFX BYTE PUBLIC,
                                                                                                                                                                                                                MSGDATAX BYTE PUBLIC,
                                                                                                                                                                                                                                                                            EXTRIND BYTE PUBLIC,
                                                                                           MSGSIM BYTE PUBLIC,
                                                                                                                                                  MSGEX BYTE PUBLIC.
                                                                                                                                                                                                                                                                                                                                                                                                                                 DBSYSTSW BYTE
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               , 32H,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      EH,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      15
```

~

```
(0E1H, 0F1H, 0C1H, 0D1H, 0E1H, 33H, 33H, 0FBH, 0, 0, 0, 0, 0, 0),
                                                                                                                                                                                                          STRCK:
                                                                                                             /* EXECUTION THBLE FOR REPLACED INSTRUCTION */
                                                                                                                                                                                                          Ξ
                                                                                         SNEXTBLA (7) ADDRESS PUBLIC AT (. SNEXTBL(0)),
8DCH, BECH, BFCH, BCDH),
SNDECTBL (5) BYTE PUBLIC INITIAL ('RMDAB'),
                                                                                                                                                                                                           ز
                                                                                                                                                                                                                                                                                                                      SNREG (12) BYTE PUBLIC,
SNREGA (6) ADDRESS PUBLIC AT (. SNREG(0)),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   /* TRUE - TYPE REGISTER CONTENTS */
                                                                                                                                                                                                                                                                           SNAPSHOT(S) NOT TYPED ()
                                                                                                                                                                                                          ï
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               - SNAPSHOT IN PROGRESS */
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      /* TEMP MEMORY DUMP STORAGE */
                                                                                                                                                                                  SNTEXT3 (66) BYTE PUBLIC INITIAL
                                         SNEXTBL (14) BYTE PUBLIC INITIAL
                                                                                                                                       SNTEXT1 (25) BYTE PUBLIC INITIAL
                                                                                                                                                                                                                                                   SNTEXT4 (28) BYTE PUBLIC INITIAL
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        /* TEMP VARIABLE STORAGE */
                                                                                                                                                                                                                                                                                                                                                                   /* TEMP REGISTER STORAGE */
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 - TYPE DUMP */
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  SNDUMP (28H) BYTE PUBLIC,
                                                                                                                                                                                                                                                                                                 SNCLEARBEG BYTE PUBLIC,
                                                                                                                                                                                                                                                                                                                                                                                          SNVAR (5) STRUCTURE (
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     CONTB BYTE> PUBLIC.
                                                                                                                                                            (B, 'SNAPSHOT AT
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           DUMPFL BYTE PUBLIC,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       SNAPFL BYTE PUBLIC,
                                                                                                                                                                                                          ..
60
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             REGFL BYTE PUBLIC,
                                                                                                                                                                                                                                                                                                                                                                                                                                       BYTETYPE BYTE,
                                                                                                                                                                                                                                                                                                                                                                                                                                                               CONTA ADDRESS,
                                                                                                                                                                                                                                                                                                                                                                                                                    ADR ADDRESS,
                                                                                                                                                                                                          (8, 6, 7 H:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              /* TRUE
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 /* TRUE
                                                                                                                                                                                                                             ?
                                                                                                                                                                                                                                                                            (4, 8, '
                                                                                                                                                                                                                                                                                                    ಶ್ವ
```

00

```
(SNB1, SNB2) BYTE PUBLIC, (SNA1, SNA2) ADDRESS PUBLIC,
                                                                                                                                                                                                                                                     /* CONDITION ADDRESS AND CONTENTS */
                                                               (SNADRCONT BASED SNADR) (1) BYTE,
/* SNAPSHOT ADDRESS AND CONTENTS */
                                                                                                                                                                                                                                                                                                                                                                                                        /* OVERLAYS FOR INT 8 AND INT 5 */
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 /* NUMBER OF LOCATIONS TO DUMP */
                     /* TRUE - CONDITION SPECIFIED */
                                                                                                                                                                                                                                                                                                                                         /* TEMP STORAGE FOR 8688-8887 */
                                                                                                                                                                                                                                                                                                                    TEMPINTO (8) BYTE PUBLIC AT (30H),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       COUMPDATA BASED DUMPBEG) (1) BYTE.
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         /* SNAP INT WORK VARIABLES */
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                (SNCONTADR BASED SNA1) ADDRESS,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       (SNSTACK BASED SNA2) (12) BYTE,
                                                                                                                            /* LAST BYTE IN MSGDATA */
SNVARX BYTE PUBLIC,
                                                                                                                                                                                                                                  (SNCOND BASED SNCONDADR) BYTE,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   (SNCONTBYTE BASED SNA1) BYTE.
                                                                                                                                                                                                                                                                                                                                                                                                                                                 /* INSTRUCTION LENGTH */
                                                                                                                                                                                                                                                                                                                                                               (MEMINTO BASED A1) (1) BYTE,
                                                                                                                                                                                                                                                                                                                                                                                  (MEMINTS BASED A2) (1) BYTE,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          /* OVERLAY FOR STACK */
                                                                                                                                                                                                                                                                                              /* SNAPSHOT COUNTER */
                                                                                                                                                                   /* INDEX FOR SNVAR */
                                                                                                                                                                                         SNCONDADR ADDRESS PUBLIC.
                                                                                                                                                                                                             SNCONDSRVE BYTE PUBLIC.
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              SNINUMDUMP BYTE PUBLIC,
                                         SNADR ADDRESS PUBLIC,
                                                                                                        MSGDATL BYTE PUBLIC,
CONDFL BYTE PUBLIC.
                                                                                                                                                                                                                                                                                                                                                                                                                             INSTRL BYTE PUBLIC,
                                                                                                                                                                                                                                                                            SNCTR BYTE PUBLIC.
                                                                                                                              11 11 11
                                                                                                                                                                                          11 11
                                                                                                                                                                                                                                   . . . . . .
                                                                                                                                                                                                                                                                                                                                                           11
                                                                                                                                                                                                                                                                                                                                                                                   11
                                                                                                                                                                                                                                                                                                                                                                                                           # #
                                                                                                                                                                                                                                                                                                                                                                                                                                                11
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             11
```

SNCLEAREND BYTE PUBLIC:

### PL/M-88 COMPILER

*/ DBSTART: PROCEDURE;;	4	128
** ***********************************		
SYSTEM START		
DESTART		
**		
*************************************		
*************************************		
*		
END;	N	119
DBEXTRACT: PROCEDURE EXTERNAL;	7	118
EMD;	N	117
SNAPTERM: PROCEDURE EXTERNAL;	1	116
END ;	N	115
SNAPINT: PROCEDURE EXTERNAL;	7	114
END;	N	113
SNAPSHOT: PROCEDURE EXTERNAL;	7	112
END;	N	111
DBTERMPER: PROCEDURE EXTERNAL;	स	110
END;	N	169
DBEXTREQ: PROCEDURE EXTERNAL;	4	168
END;	N	187
DBREG: PROCEDURE EXTERNAL;	7	186
END;	N	165
DBINPUT: PROCEDURE EXTERNAL;	7	164
<b>≴</b> EJECT		

IF DBFUNCTION = 4 THEN CALL SNAPTERM: /\* RESTART, RESTORE ACTIVATED SNAPSHOT \*/

N

DO; CALL ILLEGALMSG; RETURN; END; B1 = MSG(MN) - 20; DO CRSE B1; A* REPAICH TO MSG PROCESS */	CALL DBINPUT; CALL DBINPUT; CALL ILLEGALMSG; DO; DEBUG = FALSE; CRTINPUT = INITDEBUG; DBFUNCTION = 0FFH; IF PERFUNCTION THEN CALL DBTERNPER; END; CALL DBINPUT; CALL DBTERNPER; DO;	IF TERNPERFUNCT OR DBFUNCTION = 0FFH THEN RETURN;  A* FUNCTION NO LONGER ACTIVE */  DBTBCODE = MSGDATA(0);  CALL DBINPUT;  END;  DO;  IF DBEXTRFL THEN DBMSGCOUNT = DBMSGCOUNT + 1;  ELSE CALL DBEXTRACT;  A* EXTRACTION MSG FROM EXEC */  END;  HARDCOPY = FALSE;  END;
ammmaa	WWW44444WWWW	14 4444W44 4WW
141 142 143 145 145 146	144 148 148 148 148 148 148 148 148 148	160 160 160 160 160 160 170 170

### PL/M-80 COMPILER

RETURN	MSGENT07;	
RETU	END	DEMOD1:
		END
8	8	4
173	174	175

### MODULE INFORMATION:

= 9312H = 6664H	CODE AREA SIZE	II	= 6168H	264D
MAXIMUM STACK SIZE = 0004H 684 LINES READ	VARIABLE AREA SIZE	H	9312H	286D
684 LINES READ	MAXIMUM STACK SIZE	11	8884H	5
	684 LINES READ			
0 PROGRAM ERROR(S)	6 PROGRAM ERROR(S)			

# END OF PL/M-88 COMPILATION

ISIS-II PL/M-80 V3.0 COMPILATION OF MODULE DBMOD2 NO OBJECT MODULE REQUESTED COMPILER INVOKED BY: PLM80 :F1:DB2.SRC NOOBJECT PAGEWIDTH(80) PAGELENGTH(35)

######################################	DEMOD2: DO; \$ INCLUDE<:F1:DBDATE.SRC>	*/	:	·	**	** DEBUG MODULE	**	,并不不会不会不会不会不会不不不不不不不不不不不不不不不不不不不不不不不不不不	**	** MODULE IDENTIFICATION : DB	** MODULE NUMBER : 87	**	:	<b>,</b>	* * NOLIST	DBINPREQ: PROCEDURE EXTERNAL;	END;	DBSYNCPR: PROCEDURE EXTERNAL:	END;	DEREG: PROCEDURE EXTERNAL,	END;	DBTERM: PROCEDURE EXTERNAL:	EMD;	DEMSX: PROCEDURE EXTERNAL;	END;
# # # # # # # # # # # # # # # # # # #		11	11	11	11	11	11	11	11	11	II	H	11	11	11	н	N	ਜ	Ø	स	Ø	स	M	त्त	Ø
	#															164	165	186	167	168	109	116	111	112	113

N

DBILLEGAL: PROCEDURE EXTERNAL; END; DBINSPECT: PROCEDURE BYTE EXTERNAL; END; DBCHANGE: PROCEDURE EXTERNAL; END;	DBDUMP: PROCEDURE EXTERNAL;  DBSYSTEST: PROCEDURE EXTERNAL;  DBHARDCOPY: PROCEDURE EXTERNAL;  DBSNAP: PROCEDURE EXTERNAL;  DBSNAP: PROCEDURE EXTERNAL;	/* ***********************************	******* INPUT:	DBFL = 1;
न्यस्यस्य	स्थस्थस्थस्थ		H 0 0M	12
444444 444444 4444444	8 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2		8 5 5 E E E E E E E E E E E E E E E E E	132

RETURN; END; DO CASE CRIINPUT; GO TO DEBUGINIT; GO TO SELCPTR; GO TO SELFUNC; GO TO PROCESSFUNC; END;	/* ***********************************	DEBUGINIT: DBCRT = CS; A1 = SELCP1(A):	1 1 2		B1 = FALSE; CALL DBINPREQ; /* PFG(FST TAPUT WITHOUT PO) */	CRINPUT = SELECTOPIR;  A* PROCESS SELECT COMPUTER INPUT NEXT CRLL */	RETURN;	Antherstandingschaftertanderstandingschaftertandingschaftertanderstandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschaftertandingschafter	SELCPTR:
ммиммимм		or o	101	N (	oron	Ø	Ø		Ø
######################################		144	143	44 .	145 146	147	148		149

B3 = NSGDATA(0) - 30H;  /* DECODE CPTR NUMBER */ IF NSG(NL) () 5 OR B3 > MAXCPTR THEN D0;	7* ILLEUAL IMPOL */ A1 = .SELCP2(0);	B1 = LEMSTH(SELCP2); /* TYPE /?-CPTR:/ ON CRT */	GO TO DEINFL	END,	ELSE DO:	7* IGNUT U.K. */ IF 83 = CPTR UR 83 = 0	THEN CALL DOREGO	/* DEBUG IN OWN COMPUTER */	ELSE DO;	/* DEBUG IN OTHER COMPUTER */	EXTOB = 8;	00 82 = 1 TO 83;	EXTUB = EXTOB + MAXMOD,	EVD)	EXTOB = EXTOB - 1;	A COMPUTE MODULE NUMBER OF DEBUG MODULE IN	REQUESTED COMPUTER */	IF NOT ACTIVE(EXTDB)	THEN DO,	/* REQUESTED DEBUG MODULE NOT ACTIVE */	B1 = LENGTH(NOTACTMSG);	ht = .NOTACTMSG(0);	CALL DESYNOPRY	/* TYPE NOT ACTIVE MSG */	CALL DETERMS
61	M	M	M	(~)	N	M			M		4	4	i)	ιņ	4			4			10	in	i)		כוו
150	152	10	154	155	156	157			159		150	161	162	163	164			165			167	168	169		176

A* TERNINATE DEBUG */ RETURN	EMD;	ELSE DO:	/* REQUESTED DEBUG MODULE ACTIVE */	Danaz4(a) = EXTDB;		/* SEND DEBUG REQUEST MSG TO EXT. DEBUG MO	DULE	DATA BYTE IS MODULE NUMBER OF CRT MODUL	, * m	CRIINPUT = INITOEBUG;	/* RESET CONTROL VARIABLE */	ENO	ENED	ΞNĎ,	RETURN	**	水水水水水水水水水水水水水水水水水水水水水水水水水水水水水水水水水水水水水水水	*******	**	** SELECT FUNCTION	*	SELFUNC:	TERMPERFUNCT = FALSE,	IF MSG(ML) > 4	THEN DO:	/* CHARACTERS ENTERED */	IF MSGDATA(0) = 'P'	THEN 81 = 1;	/* PERIODIC FUNCTION */	$ELSE \; B1 = G$
เก	(i)	4		IO.	L)					in		(i)	.1	M	Ø							Ø		Ø			M			M
171	172	173		174	175					177		178	479	183	181							182		183			185			187

ø

/* NON PERIODIC FUNCTION */ B3 = MSGDATA(B1);	DO B2 = $0$ TO LAST(FUNCTBL); /* SEARCH FOR COMMAND */	IF FUNCTBL(B2) = B3 THEN	DQ;	DEFUNCTION = B2:	/* SET FUNCTION IDENTIFIER */	/* PROCESS FUNCTION INPUT NEXT */	IF FUNCTBL(B2) <> 'C' THEN DBFUNCTSAVE = 0FFH;	GO TO PROCESSFUNC:	END;	END;	END;	ELSE DO;	/* INPUT CR ONLY */	IF DBFUNCTSAVE < 3 THEN	/* INSPECT NEXT ADDRESS/BYTE */	DQ;	B3 = DBINSPECT:	RETURN	END;	IF DBFUNCTSAVE = 5 AND MSGSIM THEN	/* REPERT MSG SIMULATION */	00;	CALL DEMSX;	RETURNS	END;	END;	CALL DEILLEGAL;	/* NO CHARACTERS ENTERED
٣	M	4	4	5	u	,	ທ	I)	ريا ريا	4	M	01		M		M	4	4	4	M		M	4	4	4	M	N	
188	189	198	191	192	60.	133	194	196	197	138	199	200		201		282	203	264	202	286		282	288	283	218	211	212	

1	-
1	2
COMMISSION	
0	5

213 2 RETURN	*	**************************************	**	** PROCESS FUNCTION INPUT	/*	214 2 PROCESSFUNC:	DO CASE DEFUNCTIONS	/* BRANCH TO SELECTED FUNCTION */	215 3 IF NOT DBINSPECT THEN GO TO SELFUNC;	IF NOT DBINSPECT THEN GO TO SEL	CALL DBCHAMGE;	28 3 CALL DEDUMP;	221 3 CALL DBSNAP;	22 3 CALL DBMSX;	223 3 CALL DBHARDCOPY;	224 3 CALL DBSYSTEST;	
		*****************						7* NOI10	SELFUNC	SELFUNC							

### MODULE INFORMATION:

CALL DBTERM: END: RETURM: END DBINPUT: END DBMOD2:

544D	98	8
8228H	навав	<b>6662H</b>
11	H	11
	SIZE	SIZE
SIZE	RREA	STACK
HREA	VARIABLE	_
BOOD	VARI	MEXIMU

### PL/M-80 COMPILER

ISIS-II PL/M-80 V3.0 COMPILATION OF MODULE DBMOD3 NO OBJECT MODULE REQUESTED COMPILER INVOKED BY: PLM30 :F1:D83.5RC NOOBJECT PRGEWIDTH(80) PRGELENGTH(35)

DBNOD2: DO;	*	本书表本本本本本本本本本本本本本本本本本本本本本本本本本本本本本本本本本本本本	古古安全的大学的大学的大学的大学的大学的大学的大学的大学的大学的大学的大学的大学的大学的	**	** DEBUG MODULE	**	水水水水水水水水水水水水水水水水水水水水水水水水水水水水水水水水水水水水水水水	**	** MODULE IDENTIFICATION : DB	** MODULE NUMBER : 07	**	水水水水水水水水水水水水水水水水水水水水水水水水水水水水水水水水水水水水水水水	/*	\$ MOLIST	DBSYNCPRH: PROCEDURE EXTERNAL;	END;	DENEMLINEH: PROCEDURE EXTERNAL;	END;	DEBEGLINEH: PROCEDURE EXTERNAL;	END;	DBSYNCPR: PROCEDURE EXTERNAL;	END	PACKADR: PROCEDURE (P1) EXTERNAL;
	11	11	II	H	11	u	II	II	11	11	11	II	11	11		0	_	N	_	N	-	N	_
,															18	10	186	18	188	16	116	111	115

N

1LER	
COMP	
M-86	
5	

DCL P1 HDDRESS; END;	PACKBYTE: PROCEDURE (P1) EXTERNAL:	DCL P1 BYTE:	END; GETAGOR PROCEDURE BYTE EXTERNAL;		DBILLEGAL: PROCEDURE EXTERNAL:	END;	DENEMLINE: PROCEDURE EXTERNAL;	END;	DEBEGLINE: PROCEDURE EXTERNAL:	END;	DEPROMPT: PROCEDURE EXTERNAL;	END;	GETIMP: PROCEDURE BYTE EXTERNAL;	END;	*	**************************************	**************************************	**	TYPEINSP	PACK INSPECTION OUTPUT (A OR B)	**	安东安安安安安安安安安安安安安安安安安安安安安安安安安安安安安安安安安安安安安	/*	TYPEINSP: PROCEDURE PUBLIC;	DO B1 = 0 TO LAST(INSPBUE); /* PACK FIXED TEXT */	
NN	4	N	N +	N	7	N	7	N	7	N	4	N	7	N										4	N	
113	115	115	117	113	128	121	122	123	124	125	126	127	128	129										138	131	
										_	_															

M

DBOUTBUF(B1) = INSPBUF(B1); END;	B1 = 5;	CALL PACKADR (DBADR);	IF NOT BYTEMODE	THEN DO:	\* <b>u</b> */	DB0UTBUF(4) = 'A';	B1 = 12;	CALL PACKADR(CONTADR);	END;	ELSE DO;	/* B */	DB0UTBUF(4) = 'B';	B1 = 12;	CALL PACKBYTE(CONTBYTE);	END;	IF DETELLBACK	THEN A1 = . DBOUTBUF(0);	ELSE A1 = . DBOUTBUF(1);	CALL DESYNCPRH;	RETURN	END TYPEINSP;	*	安全在安全的全部的全部的大学的大学的大学的大学的大学的大学的大学的大学的大学的大学的大学的大学的大学的
MM	0	2	7			M	M	M	M	0		M	M	M	M	N		2	8	2	N		
132																							

DECHANGE

PROCESS CHANGE COMMAND

\*\*

4

DBCHANGE: PROCEDURE PUBLIC:	IF DBFUNCTSAVE > 1 THEN /* CHRNGE AFTER INSPECT ONLY */	DO:	CRINPUT = FUNCTIONINPUT;	GO TO DECHG1;	END;	B1 = B1 + 1;	IF NOT GETINP THEN GOTO DECHG1; /* ILLEGAL INPUT */	IF NOT BYTEMODE THEN COMISON = DRIN:	/* CHANGE ADDRESS VALUE */	ELSE DO;	/* CHRNGE BYTE VALUE */	IF DEINL (> 0 THEN GOTO DECHG1)	/* ILLEGAL INPUT */	CONTBYTE = DBINH;	END;	CALL TYPEINSP;	/* TYPE ADDRESS AND NEW CONTENTS */	CALL DBPROMPT;	/* SEND PROMPT CHARACTER */	RETURN	DBCHG1: CALL DBILLEGAL;	RETURN;	END DECHANGE:	*	*************************************	老爷爷爷爷爷爷爷爷爷爷爷爷爷爷爷爷爷爷爷爷爷爷爷爷爷爷爷爷爷爷爷爷爷爷爷爷爷爷
+	N	Ø	M	M	M	N	N	N		Ø		M		M	M	N		Ø		N	N	N	N			
153	154	155	156	157	158	159	160	162		164		165		167	168	169		176		171	172	173	174			

\*

#### DBINSPECT

## PROCESS INSPECT COMMANDS ( A AND B )

		**
175	н	DBINSPECT: PROCEDURE BYTE PUBLIC;
176	Ø	IF PERFUNCTION THEN
177	N	00;
		/* PERIODIC INSPECT */
178	М	IF NOT DETELLBACK THEN RETURN TRUE;
		/* CURRENT TYPING NOT COMPLETED YET */
189	M	CALL DEBEGLINEH;
		/* CURSOR TO BEGIN OF INPUT LINE */
181	M	CALL TYPEINSP;
		/* TYPE CONTENTS */
182	Μ	RETURN TRUE;
183	M	END;
184	Ø	IF MSG(ML) = 4 THEN
185	Ø	00;
		/* INPUT CR, INSPECT NEXT ADDRESS */
186	M	IF NOT BYTENODE
188	M	ELSE DEADR = DBADR + 1; /* INSPECT B */
189	Μ	GO TO INSP1;
196	M	END;
191	Ø	IF DBFUNCTSAVE <> OFFH THEN RETURN FALSE;
		/* NEW FUNCTION INPUT */
193	N	PERFUNCTION = B1,
194	Ø	B1 = B1 + 1,

w

#### PL/M-80 COMPILER

IF NOT GETADR THEN  /* ILLEGAL ADDRESS INPUT */	DO	CALL DBILLEGAL;	DBFUNCTSRVE = 0FFH;	PERFUNCTION = FALSE;	RETURN TRUE;	END;	DBFUNCTSAVE = DBFUNCTION:	BYTEMODE = DBFUNCTION;	IF PERFUNCTION THEN	00;	CALL DBNEWLINEH;	/* CURSOR AT BEGIN OF NEW LINE */	DBTELLBACK = TRUE;	/* REQUEST TELLBACK MSG AFTER TYPING */	CALL TYPEINSP;	RETURN TRUE;	END;	INSP1: CALL TYPEINSP;	/* TYPE REQUESTED CONTENTS */	CALL DBPROMPT;	/* SEND PROMPT CHARACTER */	RETURN TRUE;	END DBINSPECT;	END DBMOD3;	
Ø	N	Μ	M	M	М	Μ	N	N	Ŋ	N	M		M		M	M	M	N		N		N	N	ત	
195	196	197	198	199	288	261	282	203	264	202	286		287		268	289	216	211		212		213	214	215	

MODULE INFORMATION:

CODE AREA SIZE = 0168H

360D

4

# ISIS-II PL/M-80 V3.0 COMPILATION OF MODULE DEMOD4

PLM88 :F1:DB4. SRC NOOBJECT PAGEWIDTH(80) PAGELENGTH(35) NO OBJECT MODULE REQUESTED COMPILER INVOKED BY: PLMS6

.... 84 87 MODULE IDENTIFICATION MODULE NUMBER PROCEDURE (P1) EXTERNAL, DEBUG MODULE DBSYNCPŘH: PROCEDURE EXTERNAL; DBNEWLINEH: PROCEDURE EXTERNAL; DBBEGLINEH: PROCEDURE EXTERNAL; PROCEDURE EXTERNAL DBMOD4: D0; ★ INCLUDE(:F1:DBDATE. SRC) P1 ADDRESS; END; END; DBSYNCPR: END PACKADR: # NOLIST 11 11 11 **નાતાનાતાનાતાનાતા** 7 105 106 107 108 163 116 111 112

N

END; PACKBYTE: PROCEDURE (P1) EXTERNAL; DCL P1 BYTE; END; DBNEWLINE: PROCEDURE EXTERNAL; FND;	DBBEGLINE: PROCEDURE EXTERNAL; END; GETINP: PROCEDURE BYTE EXTERNAL; END; DBILLEGAL: PROCEDURE EXTERNAL; END; DBPROMPT: PROCEDURE EXTERNAL; END;	/* ***********************************	*/ DUMPLINE: PROCEDURE PUBLIC;	A2 = DUMPBEGTMP; B1 = 5;	CALL PACKADR(DUMPBEG); /* TYPE ADDRESS OF FIRST CONTENTS ON LINE */	B1 = 12;
040040	<b>u</b> a a a a a a a		4	NN	N	N
444444	122 123 123 124 126 126 126 127		128	129	131	132

M

```
INITEUMP
                       CALL PACKBYTE(DUMPCONTB(B2));
                                DUMPBEGTMP = DUMPBEGTMP + 13
                                                         CALL PACKADR (DUMPCONTA (B2));
                                                                 DUMPBEGTMP = DUMPBEGTMP + 2;
                                                                                  DBOUTBUF(B1) = ' ', B1 = B1 + 1;
                                                                                         IF DUMPBEGTMP > DUMPEND THEN
                                                                                                  /* END OF DUMP */
DO B2 = 0 TO ENDLINE;
IF BYTEMODE
                                                                                                                   DUMPDONE = TRUE;
                                                                                                                                                             END DUMPLINE;
                                                  ELSE DO;
                THEN DO:
                                                                                                                            RETURN
                                                                          END
                                          END
                                                                                                                                    END
                                                                                                                                                    RETURN
                                                                                                                                             END
                                                                                                            W 4
                                                                                                                             4 4 W U U
                                                                   4
                                                                           4 MM
 ##
##
                                         138
139
                                                                                                            146
                                                                                                                             148
                                                                                                                                     149
                                                                                                                                             158
                         136
                                                          140
                                                                   141
                                                                           142
                                                                                   143
                                                                                           145
                                                                                                                                                    151
```

INITIALIZE DUMP

PROCEDURE PUBLIC;

INITOUMP

H

153

The same of the sa

DUNPDONE = FALSE; DO B2 = 0 TO LAST(INSPBUF); /* PRESET BUFFER */	DBOUTBUF(B2) = INSPBUF(B2); END;	IF SNAPFL THEN DUMPBEGTMP = .SNDUMP(0); /* CALLED FROM SNAPSHOT */	ELSE DUMPBEGTMP = DUMPBEG; /* ORDINARY DUMP */	IF BYTEMODE THEN DO;	ENDLINE = 0FH;	DBOUTBUF(4) = 'B';	EMD;	ELSE DO;	ENDLINE = 7;	DB0UTBUF(4) = 'A';	END;	RETURN;	END INITOUMP;	*
NN	мм	N	N	N	M	M	M	N	M	M	M	Ø	N	
154 155	156 157	158	160	161	163	164	165	166	167	168	169	170	171	

DUMP CONTENTS FROM DUMPBEG TO DUMPEND MODE (A OR B) IN BYTEMODE

PUMP

\*

\*\*

\*

n

DUMP: PROCEDURE PUBLIC;	CALL DUMPLINE; IF DBTELLBACK THEN A1 = DBOUTBUF(8);	ELSE A1 = .DBGUTBUF(1); B1 = B1 - 1;	CALL DBSYNCPRH;  /* TYPE 1 ROW */  IF DIMPOONE THEN RETURN;	DUMPBEG = DUMPBEG + 10H;  /* COMPUTE BEGIN ADDRESS OF NEW LINE */	RETURN; END DUMP;	/* ***********************************	DECODE DUMP INPUT, SET DUMPBEG, DUMPEND, BYTEMODE ** *********************************	DUMPPECODE: PROCEDURE BYTE PUBLIC:	B1 = B1 + 1, BYTEMODE = MSGDATA(B1) - 'A', IF PUTEMODE > 1 THEN PETHEN EQUES:	B1 = B1 + 1; IF NOT GETINP THEN RETURN FALSE; /* ILLEGAL ADDRESS INPUT */
н	NN	NN	a a	N	N N		,	स (	NNO	100
172	173	176 177	178	181	182 183		,	184	185 186 187	189 198

ø

DUMPBEG = DBIN;  B3 = MSGDATA(B1);  B1 = B1 + 1;  IF NOT GETINP THEN RETURN FALSE;  /* ILLEGAL SECOND ADDRESS INPUT */  THEN DUMPFND = DRIN;	ELSE IF B3 = ','  THEN D0;  IF NOT BYTEMODE THEN DBIN + DBIN;  END;  ELSE RETURN FALSE;  END DUMPDECODE;	**************************************	DBDUMP: PROCEDURE PUBLIC:	IF PERFUNCTION THEN /* PERIODIC DUMP ACTIVE */	DO; IF NOT DBTELLBACK THEN RETURN; /* END OF DUMP */
00000	N MMMNNN		ત	N	NM
192 193 194 195	261 261 263 264 265 265 266		268	289	216

CALL DEBEGLINEH; /* CURSOR TO BEGIN OF LINE */	A2 = DUMPBEG; DUMPBEGTMP = DUMPBEG;	CALL DUMP;	RETURN	END;	IF DETELLBACK THEN GO TO DUMPCONT;	/* DUMP ALREADY ACTIVATED, CONTINUE */	PERFUNCTION = B1;	IF NOT DUMPDECODE THEN	/* ERROR IN DUMP INPUT */	DO;	CALL DBILLEGAL;	PERFUNCTION = FALSE;	RETURN	END;	DBFUNCTSRVE = DBFUNCTION;	B3 = DUMPEND - DUMPBEG;	IF 83 > 0FH THEN DBTELLBACK = TRUE;	/* DUMP MORE THRM 1 ROW */	IF PERFUNCTION THEN	00;	IF 83 > 0FH THEN DUMPEND = DUMPBEG + 0FH;	/* ADJUST DUMP REGION IF TOO LARGE */	DBTELLBACK = TRUE;	END;	CALL INITDUMP;	DUMPCONT: ,	CALL DENEWLINEH;	CALL DUMP;	IF PERFUNCTION THEN RETURN:	
м	M	M	M	м	N		N	Ø		N	M	M	M	M	N	N	Ø		Ø	N	M		M	M	N	N	N	N	Ø	
213	214	216	217	218	219		221	222		223	224	225	226	227	228	553	236		232	233	234		236	237	238	239	240	241	242	

ω

#### PL/M-80 COMPILER

		INPCT			
		DEBUG			
RETURNS		OR NEW			
NE THEN		PROMPT !			
IF NOT DUMPDONE THEN RETURN:	DBPROMPT	* SEND	RN	END DBDUNP;	
N FI	CALL		RETURN	END	END DBM004;
					EZP
N	N		01	c)	स
244	246		247	248	249

\*

### MODULE INFORMATION:

5810	8	9		
= 6245H	GGGGH	вваен		
11	11	11		
CODE AREA SIZE	VARIABLE AREA SIZE	MAXIMUM STACK SIZE	775 LINES READ	8 PROGRAM ERROR(S)

## END OF PL/M-80 COMPILATION

PAGE

H

ISIS-II PL/M-80 V3.0 COMPILATION OF MODULE DEMODS NO OBJECT MODULE REQUESTED

COMPILER INVOKED BY: PLM80 :F1:DB5.SRC NOOBJECT PAGENIDTH(80) PAGELENGTH(35)

.... MODULE IDENTIFICATION PROCEDURE (P1) EXTERNAL; PROCEDURE (P1) EXTERNAL; DEBUG MODULE DBNEWLINEH: PROCEDURE EXTERNAL; PROCEDURE EXTERNAL, MODULE NUMBER # INCLUDE(:F1:DBDATE, SRC) P1 RDDRESS ĝ DBSYNCPRH: F PACKBYTE: PACKADR: TSIJON 3 DBM005: 11 11 11 11 11 11 11 4040400H 184 185 186 186 168 116 H

N

v	
Щ	
ILER	
2	
8	
F	
Ĺ	
_	

DCL P1 BYTE; END;	DUMP: PROCEDURE EXTERNAL;	END;	DUMPDECODE: PROCEDURE BYTE EXTERNAL;	END;	GETINP: PROCEDURE BYTE EXTERNAL;	END;	DBSYNCPR: PROCEDURE EXTERNAL;	END;	DBILLEGAL: PROCEDURE EXTERNAL;	END;	DENEMLINE: PROCEDURE EXTERNAL;	EMD;	INITDUMP: PROCEDURE EXTERNAL;	EMD;	*	******************	**************************************	**	SNAPSHOT	PROCESS SNAPSHOT (PRIORITY PROCEDURE )	**	*************************************	/*	SNAPSHOT: PROCEDURE PUBLIC;	IF SNFL = 0 THEN	/* FIRST CALL, FIND ADDRESS OF SNAPSHOT */	DO;
00	त	N	4	N	त	N	त	N	त	N	Ŧ	Ø	7	N										त	0		N
112	114	115	116	117	118	113	128	121	122	123	124	125	126	127										128	129		138

SNAPSHOTADR = PROCADR;	SWFL = 1;	RETURN	END;	IF DBTELLBACK THEN	DO CASE DBTBCODE;	/* PROCESS TELLBACK MSG */	GO TO TBØ;	GO TO TB1;	10	G0 T0 TB3;	GO TO TB4;	END;	CALL DBNEWLINEH;	DO $B3 = 0$ TO LAST(SWIEXT1);	/* PACK HERDER */	DBOUTBUF(B3) = $SNTEXT1(B3)$ ;	END;	B1 = 13;	CALL PACKADR(SNADR);	/* PACK SNAPSHOT ADDRESS */	B1 = 19;	DO B3 = 0 TO INSTRL - 1;	/* PACK INSTRUCTION */	CALL PACKBYTE(SNEXTBL(83+8));	B1 = B1 + 1;	END;	B1 = B1 - 1	A1 = DEOUTBUF(0);	CALL DBSYNCPRH:	11	DBTELLBACK = TRUE;
M	M	M	M	Ø	N		Μ	M	M	M	M	M	N	N		M	M	N	Ŋ		N	N		M	M	M	Ø	N	0		N
131	132	133	134	135	136		137	138	139	146	141	142	143	144		145	146	147	148		149	159		151	152	153	154	155	156		157

IF REGFL THEN  /* TYPE REGISTER CONTENTS */  DO,  CALL DBNEWLINEH;  DO B3 = 0 TO LAST(SNTEXT3);  DBOUTBUF(B3) = SNTEXT3(B3);  END;	B1 = 6; CALL PACKBYTE(SNREG(1)); /* PACK A */ B1 = 11; D0 B3 = 2 T0 7 BY 2; /* PACK B, C, D, E, H, L */	CALL PACKBYTE(SNREG(B3+1)),  B1 = B1 + 3;  CALL PACKBYTE(SNREG(B3)),  B1 = B1 + 3;  END;  CALL PACKBYTE(SNREG(10));  /* PACK N */  B1 = B1 + 7;	CALL PACKADR(SNREGA(4)),
U UWW44	MM MM	4444W W	м ммм мааам
158 159 161 161 163	164 165 166 167	168 169 171 172 172 173	175 176 177 178 189 183 183

/* PRCK SIGN */	000	DBOUTBUF(81) = $^{<5}$ ; B1 = B1 + 2;	END;	IF (B3 AND 01000000B) <> 0 THEN	000	DBOUTBUF(B1) = $(2')$ :B1 = B1 + 2;	END;	IF (B3 AND 4) <> 0 THEN	/* PACK PARITY */	;0d	DEGUTBUF(B1) = $'P'$ ; B1 = B1 + 2;	END	IF (B3 AND GGG1GGGGB) <> @ THEN	/* PACK AUX. CARRY */	DQ;	DBOUTBUF(B1) = $'R'$ ; DBOUTBUF(B1+1) = $'C'$ ; B1 = B1 + 2;	END;	A1 = . DEGUTBUF(0);	CALL DBSYNCPRH;	/* TYPE AND CONTINUE AT TB0 WHEN TYPING COMPLETED */	RETURN	END;	TBØ: IF SNVRRX <> 0 THEN	/* TYPE REQUESTED VARIABLES */	DO;	CALL DENEWLINEH;	DBOUTBUF( $\theta$ ) = 1; DBOUTBUF(1) = $\theta$ ;	00 B3 = 2 T0 65	DBOUTBUF(B3) = ' ';	EMD;
	M	4	4	м	M	4	4	M		M	4	4	M		١٣١	4	4	M	M		M	M	N		01	M	M	M	4 .	4
	185	186	188	189	190	191	193	194		195	196	198	199		200	201	264	285	286		287	268	209		210	211	212	214	212	216

B1 = 5;	DO B3 = 0 TO SNVARX-1;	/* TYPE CONTENTS OF VARIABLES */	CALL PACKADR(SNVAR(B3), ADR);	DBOUTBUF(B1) = $':'$ ; B1 = B1 + 2;	IF SWYR(B3), BYTETYPE	THEN DO;	/* BYTE VARIABLE */	DB0UTBUF(81-7) = $'8'$ ;	CALL PACKBYTE(SNVAR(B3), CONTB);	END;	ELSE DO;	/* ADDRESS VARIABLE */	DB0UTBUF(81~7) = 'A';	CALL PRCKADR(SWVRR(B3), CONTA);	END;	B1 = B1 + 2:	END;	A1 = DBOUTBUF(8);	CALL DESYNCPRH;	/* TYPE AND CONTINUE AT TB1 WHEN TYPING COMPLETED	RETURN:	END;	SNA1 = DUMPREG;		/* NO DUMP REQUESTED */	CALL INITDUMP;	CALL DENEMLINEH;	DBOUTBUF( $\theta$ ) = 2;	CALL DUMP;	RETURN	IF NOT DUMPDONE THEN
																							TB1:								TB2:
M	M		4	4	4			ທ	in)	เก	4		ı,	ı,	I)	4	4	M	M		M	M	7	N		7	7	0	N	7	N
217	218		219	226	222			224	225	226	227		228	229	230	231	232	233	234		235	236	237	238		240	241	242	243	244	245

~

DO; CALL DBNEWLINEH; DBOUTBUF(Ø) = 3; CALL DUMP; RETURN; END;	: DUMPBEG = SNR1. IF SNCTR <> 0 THEN  /* TYPE NUMBER OF SKIPPED SNRPSHOTS */  DO:	CALL DENEWLINEH;  DO B3 = 0 TO LAST(SNTEXT4);  DBOUTBUF(B3) = SNTEXT4(B3);  END;	B1 = 4; CALL PACKBYTE(SNCTR); A1 = .DBOUTBUF(0); B1 = LENGTH(SNTEXT4); CALL DBSYNCPRH; SNCTR = 0; END;	: SNAPFL = FALSE;  DBTELLBACK = FALSE;  CALL DBNEWLINEH;  RETURN;  END SNAPSHOT;	/* ***********************************
	TB3:				* *
NMMMMM	00 0	W W 4 4	M M M M M M M	00000	
245 245 249 250 250 250 250	25 25 25 25 25 25 25 25 25 25 25 25 25 2	255 256 257 258 258	259 268 261 262 263 264 264	266 267 268 269 279	

DESNAP

\*\*

œ

#### PL/M-80 COMPILER

#### INITIATE SNAPSHOT

**************************************	. 60 244 ·	93 3 SN1: DO CASE B2;	
----------------------------------------	---------------	-----------------------	--

ø

/* 'R' PARAMETER */  REGFL = TRUE;  B1 = B1 + 1;  END;  D0;  /* 'M' PARAMETER */	B1 = B1 + 1; IF NOT GETINP THEN GO TO SNILL; SNCONDADR = DBIN; /* SET CONDITION ADDRESS */	CALL IF M B1 = CALL	IF NOT GETINP THEN GO TO SNILL; IF DBIN > 0FFH THEN GO TO SNILL;  /* CONDITION NOT CORRECT */ SNCONDSAVE = DBINH;  /* SET CONDITION */	COND END; DO; /* IF N	S TI S TI OO S NO UO
N N N A	ດເທດ	เมเมเม		ທທ4 ໜ	ממממ ממ
295 296 297 298	299 366 362	363 364 366 387	368 316 312	21.5 21.5 21.5 21.5 21.5 21.5	318 319 320 321 322 323

DUMPFL = TRUE; END; GO TO SNAB; /* 'A' PARAMETER */	SNAB: DO;	IF SNVARX > LAST(SNVAR) THEN GO TO SN2; /* MORE VARIABLES THAN ALLOWED */	SNVAR(SNVARX) BYTETYPE = MSGDATA(B1) - 'A'; /* SET VARIABLE TYPE */	B1 = B1 + 1;	IF NOT GETINP THEN GO TO SNILL;	SNVAR(SNVARX), RDR = DBIN;	/* SET ADDRESS OF VARIBLE */	SWYRKX = SWYRKX + 1;	EMD;	END; /* DO CASE B2 */	SN2: B1 = B1 + 1;	END; /* DO WHILE B1 <= MSGDRTL */	B3 = SNADRCONT(0);	DO B2 = 0 TO LAST (INSTRTBL);	/* DETERMINE LENGTH OF SNAPSHOT INSTRUCTION */	IF INSTRIBL(B2) = B3 THEN  A INSTRIBL CET LENGTH #/	DO;		THEN INSTRL = 2;	ELSE INSTRL = 3;	GO TO SN3;	END;	END;
N N 4	4	ທ	W)	ທ	ທ	S		เก	เก	4	M	M	Ø	N		M	м	4		4	4	4	M
325 325 326	327	328	330	331	332	334		335	336	337	338	339	346	341		345	343	344		346	347	348	349

INSTRL = 1;	SN3: DO B2 $\approx$ 0 TO INSTRL - 1; /* MOVE CASE TACTE THIS EXECUTION TRREE $\approx$ /	SNEXTBL(82+8) = SNAPRCONT(82);	END;	SNEXTBL(11) = 0C3H;	/* INSERT JUMP INSTRUCTION */	SNEXTBLA(6) = SNADR + INSTRL;	/* SET ADDRESS OF INSTR AFTER SWAP INSTR */	SNADRCONT(0) = 0E7H;	/* INSERT RSTØ INSTR AT SNAP ADDRESS */	DBFUNCTSAVE = DBFUNCTION;	PERFUNCTION = TRUE;	CALL DBNEWLINE;	RETURN	SNILL: CALL DBILLEGAL;	RETURN;	END DESNAP;	*/	安全安全的安全的安全的安全的安全的安全的安全的安全的安全的安全的安全的安全的安全	**************************************	**	SNAPTERM
N	N	M	Μ	Ø		N		N		N	N	N	N	N	N	N					
358	351	352	353	354		355		356		357	358	359	360	361	362	363					

## TERMINATE SNAPSHOT FUNCTION

364 1 SNAPTERM: PROCEDURE PUBLIC:

DO B1 = 0 TO INSTRL-1; /\* RESET SNAPSHOT INSTRUCTION \*/

N

365

SNADRCONT(B1) = SNEXTBL(B1+8);  END;  SNAPFL = FALSE;  RETURN;  END SNAPTERN;  /*  ********************************	PROCESS SNAPSHOT INTERRUPT  IF SNAPSHOT IS ACTIVE THIS PROCEDURE IS EXECUTED WITH RST4  ** ********************************	IF SNFL = 0 THEN /* FIRST CALL */ DO; SNAPINTADE = PROCADE;	RETURN; END; IF CONDFL AND SNCONDSAVE <> SNCOND THEN GO TO INTRET;	/* SPECIFIED CONDITION IS NOT NET */ IF SNAPFL THEN /* ALREADY A SNAPSHOT IN PROGRESS */ DO; SNCTR = SNCTR + 1; GO TO INTRET; END;
MMNNN	н	0 01	1 M M N	0 0MMM
366 368 369 379	371	372	328	386 381 382 383

SNAPFL = TRUE; SNB1 = 0; IF SNVARX <> 0 THEN /* SAVE SPECIFIED VARIABLES */	DO WHILE SNB1 < SNVARX; SNR1 = SNVAR(SNB1). ADR; IF SNVAR(SNB1). BYTETYPE THEN SNVAR(SNB1). CONTR = SNCONTRYTE;	ELSE SNVAR(SNB1). CONTA = SNB1 = SNB1 + 1; END;	IF DUMPFL THEN  /* SRVE DUMP AREA */  DO SNB1 = 0 TO SNNUMDUMP;  SNDUMP(SNB1) = DUMPDATA(SNB1);  END.	IF REGFL THEN  /* SAVE REGISTERS */  DO;  SNA2 = STACKPTR;  DO SNB2 = 0 TO 7;  /* CANE PSU P. D. H */	SNREG(SNB2) = SNSTACK(SNB2); END; SNREG(8) = SNSTACK(10); SNREG(9) = SNSTACK(11); SNA1 = SNREGA(3);	SNREG(10) = SNCONTBYTE;  /* SAVE M */ END;  CALL PRIORITYINT(SNPRIORX);
000	NMM	MMM	N NM	MMN NM	4 4 W W	M M M M
385 385 386	387 388 389	391 392 393	396	398 398 466 461	462 463 464 466	468 469 416

SNR1 = . SNEXTBL(0);	CALL SNA1;  /* EXECUTE REPLACED INSTRUCTION AND RETURN TO	INTERRUPTED PROGRAM */	RETURN	END SNAPINT;	END DEMODS;
O	N		N	N	त
411	412		413	414	415

### MODULE INFORMATION:

1885D	ପ୍ର	4		
= 075DH	HOODO	<b>6664H</b>		
11	11	11		
CODE AREA SIZE	VARIABLE AREA SIZE	1ZE	968 LINES READ	6 PROGRAM ERROR(S)

## END OF PL/M-80 COMPILATION

4

1515-11 PL/M-80 V3. 0 COMPILATION OF MODULE DEMODE NO OBJECT NODULE REQUESTED

PLM88 :F1:DB6, SRC NOOBJECT PRGEWIDTH(80) PRGELENGTH(35) COMPILER INVOKED BY:

5.安全是不是不是有的,是是不是有的的,是是是不是有的,是是是有的,我们们的,我们们的,我们们的,我们们们的,我们们们们的,我们们们们们的。 .. 0E MODULE IDENTIFICATION DEBUG MODULE DBSYNCPRH: PROCEDURE EXTERNAL; DBNEWLINEH: PROCEDURE EXTERNAL; MODULE NUMBER F INCLUDE (:F1:DBDATE, SRC) TSIJON \$ DEMODE 11 11 11 11 11 11 11 40404040A 164 165 166 166

PROCEDURE BYTE EXTERNAL,

PROCEDURE EXTERNAL,

PROCEDURE EXTERNAL,

DBPROMPT: F.

GETINE

EMD;

DBSYNCPR:

168 116 111 N

END	DBILLEGAL: PROCEDURE EXTERNAL;	END;	DBNEWLINE: PROCEDURE EXTERNAL;	END;	DBINPREQ: PROCEDURE EXTERNAL;	END;	PACKBYTE: PROCEDURE (P1) EXTERNAL;	DCL P1 BYTE;	END;	*	**************************************	*************************************	**	DBMSXINP	MSG SIMULATION AND MSG EXTRACTION	**	**************************************	/*	DBMSXINP: PROCEDURE PUBLIC:	DO CASE DEMSMITCH;	/* BRANCH TO PROCESSING OF IMPUT */	GO TO DEMB;	GO TO DBM1;	GO TO DEM1;	GO TO DBM1;	GO TO DEM1;	GO TO DEM2;	END;
N	+	2	7	N	7	N	4	2	N										7	N		M	M	M	M	M	M	M
113	114	115	116	117	118	119	120	121	122										123	124		125	126	127	128	129	138	131

ER
ጟ
5
000
F
٠,

				+
IF MSGSIM AND MSG(ML) = 4 THEN GO TO DEMSEND,  /* CONTINUATION OF MSG SIMULATION, REPEAT MSG*/  MSGSIM = FALSE, MSGEX = FALSE,  B1 = B1 + 1;  DBFUNCTSAVE = DBFUNCTION;  IF MSGDATA(B1) = 'S'  THEN MSGSIM = TRUE,  /* MSG SIMULATION */	ELSE IF MSGDATA(B1) = 'X' THEN MSGEX = TRUE; /* MSG EXTRACTION */	ELSE DO; CALL DBILLEGAL; RETURN; END;	DO B3 = 0 TO LAST(DEBUGMCB);  /* RESET MCB */  DEBUGMCB(B3) = 0FFH;  END;  MSGDATAX = 0;  DBMSWITCH = 1;  GO TO DBM11;	B1 = 0; IF MSGEX AND MSG(ML) = 4  /* INPUT CR ONLY */ THEN DO; END; ELSE IF NOT GETINP OR DBIN > 0FFH OR  CDBMSWITCH = 4 AND DBIN > LENGTH(DBMSGDATA) +
DBM6:			2 233	DBM1:
0 0000	0 0	NMMM	a mmaaa	00 0
132 134 136 137 138	148	1 1 1 1 1 4 2 4 4 4 4 4 5 4 4 5 4 4 6 6 6 6 6 6 6 6 6	146 148 148 150 151	152 153 156

THEN DO:

(2,2)				SWITCH-1) = DBINS			/*						ART(0)) THEN RETURNS								>									
B1 = 2; R1 = . (8, ??');	CALL DESYNCPR	GO TO DBM11;	END;	ELSE DEBUGMCB(DBMSWITCH-1) = DBIN:	DEMSMITCH = DEMSMITCH + 15	IF DEMSMITCH = 5 THEN	/* INPUT MCB COMPLETED */	00;	IF MSGEX THEN	/* MSG EXTRACTION */	:00	DEMSWITCH = 0;	IF NOT SEND(, EXTRACTSTART(0)) THEN RETURN.	CALL DENEMLINEH;	DBEXTRFL = FALSE;	DEMSGCOUNT = 0;	PERFUNCTION = TRUE;	RETURN	END;	IF DEBUGMCB(ML) > 4	/* INPUT DATA BYTES */	THEN CALL DENEMLINE;	ELSE GO TO DEMSEND;	END;	A4 = MSGTEXTX(DBMSWITCH);	B1 ≈ BL;	A1 = . MSGTEXT(BU);	S	B1 = FALSE;	CALL DEINPREQ;
																									DEM11:			DEM3:		
M	M	M	M	N	7	7		N	M		M	4	4	4	4	4	4	4	4	M			M	M	N	7	N	0	Oi .	N
158	168	161	162	163	164	165		166	167		168	169	178	172	173	174	175	176	177	178			188	181	182	183	184	185	186	187

RETURN	DBM2: IF NOT GETINP OR DBIN > OFFH THEN  /* ILLEGAL DATA BYTE INPUT */	DQ;	B1 = 3; A1 = . (6, '?:');	GO TO DBM3;	END;	DBMSGDATA(MSGDATAX) = DBIN;	IF MSGDATAX >= DEBUGNCB(ML) - 5 THEN GOTO DBMSEND;	/* INPUT DATA BYTES COMPLETED */	MSGDATAX = MSGDATAX + 1;	B1 = 3; $R1 = .(6, ':')$ ;	GO TO DEM3;	DBMSEND: IF NOT SEND(. DEBUGNCB(0)) THEN	/* SEND MSG */	DQ;	CALL DENEMLINE;	B1 = 15; A1 = . (0, ' MSG NOT SENT');	GO TO DBMSEND1;	END;	IF DEBUGNCB(ML) > 4 THEN	/* TRANSFER DATA BYTES */	DO B3 = 0 TO MSGDATAX;	MSGDATA(B3) = DBMSGDATA(B3);	END;	CALL DENEMLINE;	B1 = 11; A1 = . (8, ' MSG SENT');	DBMSEND1: CALL DBSYNCPR;	DEMSMITCH = 0;	CALL DEPROMPT:
N	N	N	M	M	M	N	N		0	N	N	N		N	M	M	M	M	N		N	M	M	7	N	0	N	7
188	189	190	191	193	194	195	196		198	199	201	202		203	264	202	282	288	503		216	211	212	213	214	216	217	218

ω

œ
ER
IL
-
<u>a</u>
COMP
Ö
U
0
88
Ÿ
Ė
J
5

RETURN;  END DBMSXINP;  /*  ********************************	PROCESS EXTRACTION MSG FROM EXEC ** ********************************	DBEXTRACT: PROCEDURE PUBLIC;	DCL EXTRMSG (100) BYTE;	IF NOT DBTELLBACK THEN  /* FIRST CALL */	DO:	B2 = MSGDATA(ML) - 1;	/* SAVE EXTRACTED MSG */	EXTRMSG(B1) = MSGDATA(B1);	END;	DETECODE = 0;	EMD;	DO CASE DBTBCODE;	GO TO EXTRØ;	GO TO EXTR1;	GO TO EXTR2;	GO TO EXTR3;	END
NN		7	N	N	O)	MI	1	4	4	M	M	N	M	M	M	M	M
219		221	222	223	224	225	777	227	228	229	230	231	232	233	234	235	236

DBEXTRFL = TRUE;  DO B2 = 0 TO LAST(MSGTEXT);  DBGUTBUF(B2) = MSGTEXT(B2);  END;  DO B2 = 2 TO 5;  A* PACK RNN, MN, ML */  A4 = MSGTEXTX(B2);  B1 = BU - 2;  CALL PACKBYTE(EXTRMSG(B2-2));  END;  IF EXTRMSG(ML) = 4 THEN  A* NO DATH BYTES */  DO;  DBOUTBUF(0) = 2;  GO TO EXTR12;  END;  B1 = 34; EXTRIND = 4;  GO TO EXTR11;	: IF HARDCOPY THEN DO; CALL DBNEWLINEH; B1 = 2; END; ELSE DO; DBOUTBUF(2) = 0CH; DBOUTBUF(3) = 64; DBOUTBUF(4) = 119;  A* CURSOR TO CHARACTER POSITION 36 ON INPUT LINE *  B1 = 5; END;
EXTRØ:	EXTR4:
аамма мммма амммаа	a mmmam mm
233 233 233 244 244 244 244 244 244 244	254 256 258 258 258 268 268 263 264

EXTR11: B3 = 0;  D0 WHILE EXTRIND < EXTRMSG(ML);  /* TYPE ALL DATA BYTES */ CALL PACKBYTE(EXTRMSG(EXTRIND)); DBOUTBUF(B1) = ' ', B1 = B1 + 1;  /* TYPE 1 DATA BYTE */ B3 = B3 + 1; EXTRIND = EXTRIND + 1; IF B3 > 0FH THEN G0 TO EXTR12;  /* 10H DATA BYTES TYPED ON 1 LINE */ END;	DBOUTBUF(0) = 2;  /* SET TELLBACK CODE */  EXTR12: A1 = . DBOUTBUF(0);  DBTELLBACK = TRUE;  CALL DBSYNCPRH;  RETURN;	EXTR2: IF DBMSGCOUNT = 0 THEN GO TO EXTR3;  /* SAME MSG NOT OCCURRED DURING THIS PROCESS TIME */  CALL DBNEWLINEH;  DO B2 = 0 TO LAST(SKIPMSG);  DBOUTBUF(B2) = SKIPMSG);  B1 = 3;  CALL PACKBYTE(DBMSGCOUNT);  /* TYPE NUMBER OF SKIPPED NSG */  B1 = LENGTH(SKIPMSG); A1 = . DBOUTBUF(0);  CALL DBSYNCPRH;  DBMSGCOUNT = 0;  RETURN;
	0 0000	
265 266 267 279 271 272 272	275 276 277 278 278 279	286 283 283 284 285 285 286 286 297 297 297 297 297 297 297

Q

#### PL/M-80 COMPILER

EXTR3: DBTELLBACK = FALSE;	DBEXTRFL = FALSE;	CALL DBNEWLINEH;	RETURN	END DBEXTRACT;	*	**************************************	***************************************	**	DBMSX
N	N	N	N	N					
293	294	295	296	297					

PROCESS MS AND MX INPUTS

PROCEDURE PUBLIC:

DBMSX:

298

/\* TELLBACK MSG RECEIVED \*/ THEN CALL DBEXTRACT, ELSE CALL DBMSXINP, RETURN, END DBMSX; END DBMSX; IF DBTELLBACK N 299

#### MODULE INFORMATION:

1123D 166D = 0463H VARIABLE AREA SIZE = 8864H CODE AREA SIZE

ISIS-II PL/M-80 V3.0 COMPILATION OF MODULE DBMOD7 NO OBJECT MODULE REQUESTED

PLM80 :F1:D87. SRC NOOBJECT PAGEWIDTH(80) PAGELENGTH(35) COMPILER INVOKED BY:

#INCLUDE(:F1:DBDRTE, SRC) DBM007: H

DEBUG MODULE 11

11 11

MODULE IDENTIFICATION MODULE NUMBER 11 11

... 80 78

11

11

\$ NOLIST 11 11

DBNEWLINE: PROCEDURE EXTERNAL; PROCEDURE EXTERNAL DBSYNCPR: **ਜ਼ਗ਼ਜ਼ਗ਼ਜ਼**ਗ਼ 105 105 106 108 108 109

PROCEDURE EXTERNAL, DBINPREG: r END; EMD;

#### PL/N-80 COMPILER

DBSYSTINF

#### RETURN FALSE IF INPUT NON NUMERICAL TRANSFER INPUT INTO DESTRUF

TRUE OTHERWISE

DBSYSTINP: PROCEDURE BYTE: ત્ત 110 SYSTILL (3) BYTE DATA (1 ? 1) DO

B3 = MSGDATA(B1); B2 = MSG(ML) - 5; DO B1 = 0 TO B2;

MMBB

117 118 119 126 121 122

THEN 83 = 83 - 37HJ ELSE 83 = 83 - 7873 IF 83 >= 'A'

A1 = SYSTILL(0)Ö

IF B3 > 6FH THEN

B1 = LENGTH(SYSTILL)CALL DESYNOPRO RETURN FALSE;

4 4

DBSTBUF(DBSTBUFX) = 83

DESTRUFX = DESTRUFX + 1

4 WWW WW

128 127

124 125 126

RETURN TRUE

END DESYSTINE:

N

111

112 113 114 115 M

#### PL/11-80 COMPILER

*	
*************************************	
*************************************	
**	

DBSYSTEST

## PROCESS SYSTEM TEST INPUTS

** ***********************************	DO CASE DBSYSTSW: 60 TO SW0: 60 TO SW1: 60 TO SW2: 60 TO SW2:	END: PERFUNCTION = TRUE; DBFUNCTSAVE = DBFUNCTION;	CALL DEMEMLINE;  DBSYSTSW = 1,  A1 = .SYSTTEXT1(0);  B1 = LENGTH(SYSTTEXT1);  A* TYPE ' FUNCTION:' */  G0 T0 SW4;	IF NOT DBSYSTINP OR MSG(ML) = 4 THEN GOTO SW11; DBSYSTSW = 2; Al = .SYSTTEXI2(0);
** ********** */ DBSYSTEST:		3M8	SW02	SW1: SW11:
н	NMMMM	M NNC	NNNNN N	000
130	######################################	138	4444 4444 4444 44444 44444	244 744 248

#### PLATI-86 COMPILER

B1 = LENGTH(SYSTTEXT2); G0 T0 SU4;	IF MSG(ML) = 4 THEN GO TO SWS: $/*$ INPUT CR ONLY */ THE NOT DESPETTED THEN GOTO SUM4:		CALL DASYNCPRU B1 = FALSE CALL DAINPREQU RETURNU	IF NSG(NL) = 4 THEN GOTO SWS: IF NOT DESYSTINE THEN GOTO SW21.	SYSTMSG(NL) = DBSTBUFX + 4; IF NOT SEND(, SYSTMSG(0)) THEN GOTO SW01; 2* SEND NSG 50H TO T2 *2	B2 = D8STBUFX - 1, D0 B1 = 0 T0 B2, /* TRANSFER DATA */	MSGDATA(B1) = DBSTBUF(B1); END; GO TO SUBL; END DBSYSTEST; DBMOD?;
	SM2:	SW21:	SW4:	.: 6740	3843		END DE
ભ ભ	0 0	10000	लललल	0.01	0.01	or or	MMNNH.
149 156	151	123,52,53	159 166 161 162	163 165	167 168	176 171	172 174 174 175

ISIS-II PL/M-80 V3.0 COMPILATION OF MODULE DBMOD? NO OBJECT MODULE REQUESTED COMPILER INVOKED BY: PLM80 :F1:D87.5RC NOOBJECT PAGEWIDTH(80) PAGELENGTH(35)

#INCLUDE(:F1:DBDATE, SRC) Ö DBMOD7: ᆏ

DEBUG MODULE

MODULE IDENTIFICATION MODULE NUMBER

# NOLIST

PROCEDURE EXTERNAL, PROCEDURE EXTERNAL, DBSYNCPR: FR END: DBINPREQ: END; 404040

AD-A059 601

NAVAL POSTGRADUATE SCHOOL MONTEREY CALIF
A REAL-TIME OPERATING SYSTEM FOR SINGLE BOARD COMPUTER BASED DI--ETC(U)

UNCLASSIFIED

AGE 4

Rese 601

AGE 4

Rese 601

AGE 4

Rese 601

AGE 601

END
DATE
FILMED
-12-78

N

#### PL/M-80 COMPILER

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

DBSYSTINP

#### RETURN FALSE IF INPUT NON NUMERICAL TRANSFER INPUT INTO DESTBUF TRUE OTHERWISE

SYSTILL (3) BYTE DATA (' ? '); PROCEDURE BYTE; DBSYSTINP: 정 110

DESTBUFX = DESTBUFX + 1; DESTBUF(DESTBUFX) = B3; B1 = LENGTH(SYSTILL); THEN B3 = B3 - 37H; ELSE B3 = B3 - '0'; R1 = . SYSTILL(0); B3 = MSGDATA(B1); IF B3 > OFH THEN CALL DBSYNCPR; B2 = MSG(ML) - 5; RETURN FALSE; DO B1 = 0 TO B2; IF B3 >= 'A' RETURN TRUE; END DBSYSTINP; EZO MMMMN 117 118 119 120 121 122 123 123 124 126 126 128 128 114 111 112 113

M

#### PL/M-80 COMPILER

**************************************	**************************************	DESYSTEST: PROCEDURE PUBLIC;	۵	GO TO SM1; GO TO SM2; GO TO SM3;		SW8: PERFUNCTION = TRUE;	SW61:	DBSTBUFX = 0; DBSYSTSW = 1;	SM82:	3	SW1: IF NOT DBSYSTINP OR MSG(ML) = 4 THEN GOTO SW11: DBSYSTSW = 2;  SW11: A1 = .5YSTTEXT2(0);
		4	NM	MMM	M	N	NN	NN	00	 N	000

#### PL/M-86 COMPILER

B1 = LENGTH(SYSTTEXT2); G0 T0 SW4;	SW2: IF MSG(NL) = 4 THEN GO TO SWS; /* INPUT CR ONLY */	IF NOT DESYSTINP THEN GOTO SW11: DESYSTSW $\approx 3$ :	£	B1 = LENGTH(SYSTTEXT3); G0 T0 SW4:		81 =	CALL DBINPREQ;	RETURN	SW3: IF MSG(ML) = 4 THEN GOTO SW5;	IF NOT DESYSTINP THEN GOTO SW21,	SW5: SYSTMSG(ML) = DBSTBUFX + 4;	IF NOT SEND(, SYSTMSG(0)) THEN GOTO SW01;	/* SEND MSG 50H TO T2 */	B2 = DBSTBUFX - 1;	DO B1 = 8 TO B2;	/* TRANSFER DATA */	MSGDATA(B1) = DBSTBUF(B1);	END;	GO TO SMB1;	END DESYSTEST;	END DEMOD7;
00	N	00	N	~ ~	. 0	0	N	N	N	N	8	8		N	7		M	M	0	7	7
149 150	151	153	156	157	159	168	161	162	163	165	167	168		178	171		172	173	174	175	176

#### PL/M-80 COMPILER

ISIS-II PL/M-80 V3.0 COMPILATION OF MODULE DEMOD8 NO OBJECT MODULE REQUESTED COMPILER INVOKED BY: PLM80 :F1:D88.SRC NOOBJECT PAGEWIDTH(80) PAGELENGTH(35)

# INCLUDEC:FI:DBORTE.SRC)  - /*  - ******************************

The second secon

#### PACKBYTE

BUFFER	
OUTPUT	81
IMTO	POSITION
VALUE	RT POSI
BYTE	_
PACK	STARTING

110

PROCEDURE (BYT) PUBLIC: PACKBYTE:

BYT BYTE; ಶ N

11

CALL CONVASC(BYT); **aaaaaa** 

DEGUTEUF(81+1) = BL; DEGUTEUF(B1) = BU;

B1 = B1 + 2;RETURN

END PRCKBYTE;

PACK ADDRESS VALUE INTO OUTPUT BUFFER

PACKADR

STARTING AT POSITION B1

PROCEDURE (RDR) PUBLIC: PACKADR:

118

119

(ADRH, ADRL) BYTE AT (. ADR); ADR ADDRESS, ತ್ತ N

M

L	
7	•

CALL PACKBYTE(ADRL); CALL PACKBYTE(ADRH); RETURN; END PACKADR; /* **********************************	*** GET INPUT FROM CRT AND LEAVE IT IN A4	** ************************************	GETIMP: PROCEDURE BYTE PUBLIC;	CALL DEBLK;	IF NOT GETNUM THEN RETURN FHLSE; DBIN = 64;	RETURN TRUE;	END GETINP;	*	***************************************	**************************************	**	GE FECK
0000			4	0	NN	N	N					
121 122 122 123			124	125	126	129	130					

GET INPUT ADDRESS AND LEAVE IT IN DBADR

ILER
COMP
-80
7

GETADR: PROCEDURE BYTE PUBLIC;	IF NOT GETINP THEN RETURN FALSE;	DBADR = DBIN; /* SET ADDRESS */	RETURN TRUE;	*	**************************************	*** DBTERMPER	TERMINATE PERIODIC DEBUG FUNCTION	** ***********************************	*/ DBTERMPER: PROCEDURE PUBLIC;	IF NOT PERFUNCTION THEN	DQ;	CALL DBILLEGAL;	RETURN	END;	TERMPERFUNCT = TRUE;	PERFUNCTION = FALSE;	IF MSGEX AND SEND(.EXTRACTSTOP(0)) THEN /* MSG EXTERCTION ACTION ACTION ACTION	DQ;	MSGEX = FRLSE;	DBEXTRFL = TRUE;
त्त	N	N	00						त	Ø	N	M	M	M	N	N	N	N	M	M
131	132	134	135						137	138	139	146	141	142	143	144	145	146	147	148

END;  DBMSWITCH = 0; DBSYSTSW = 0;  IF DBFUNCTION = 4 THEN CALL SNAPTERN;  IF DBFUNCTION = 0FFH THEN RETURN;  CALL DBPROMPT;  RETURN;  END DBTERMPER;	/* ***********************************	PROCESS 'H' CONMAND ** *********************************	DCL LPINITMSG (4) BYTE DATA (LP, DB, 13, 4);	IF HARDCOPY THEN HARDCOPY = FALSE;  /* TERMINATE HARDCOPY */	ELSE DO; IF SEND(.LPINITMSG(0)> THEN HARDCOPY = TRUE; /* INITIALIZE LP */	END; CALL DBPRONPT; RETURN; END DBHARDCOPY;	ETO DEFICIOS
m a a a a a a		ત	N	a	NM	M 01 01 01 4	-
158 158 158 158 158		159	160	161	163	166 167 168 169	PLE

#### PL/M-80 COMPILER

ISIS-II PL/M-80 V3.0 COMPILATION OF MODULE DBMOD9 NO OBJECT MODULE REQUESTED COMPILER INVOKED BY: PLM80 :F1:DB9.5RC NOOBJECT PAGEWIDTH(80) PAGELENGTH(35)

**************************************
**************************************

N

DBMN16(0) = DBCRT; B1 = SEND(.DBMN16(0)); RETURN; END DBNEWLINE; /* **********************************	** POSITION CURSOR AT BEGIN OF INPUT LINE **	**************************************	DBMN17(0) = DBCRT; B1 = SEND(.DBMN17(0)); RETURN;	END DBBEGLINE; /* **********************************
0000		н	000	N
105 106 107 108		169	110 111 112	113

IF HARDCOPY = TRUE : NEW LINE ON LP FALSE: NEW LINE ON CRT

_			
ũ	ľ		
Ù	ú	ı	
	-	•	
3	Ē	١	
č	7	ì	
	1	١	
7	_		
ć	Q	1	
C	ï	ı	
1	١		
741	ŀ		
:		۰	
	1	ì	
ō	ť	۱	
•	۰	۰	

114	4	DENEMLINEH: PROCEDURE PUBLICA
115	N O	B3 = DBCRT; TE MODECCEU THEN CACET = 1 P.
118	v 17	CALL DENEMLINE;
119	N	DECRT = 83;
126	8	RETURN
121	7	END DENEMLINEH;
		*
		老者我是我的我们的我们的我们的我们的我们的的,我们的我们的我们的,我们的我们的我们的我们的,我们就会会会会会会会会会会会会会会会会会会会会会会会会会会会会会会会会会会会会
		老者我在我们的我们的我们的我们的我们的我们的的的,我们的我们的的,我们的我们的的,我们就是我们的的,我们们们们们们们们们们们的。
		**
		DEBEGL INEH
		IF HARDCOPY = TRUE : NEW LINE ON LP
		FALSE: BEGIN OF LINE ON CRT
		**
		安安华大学大学大学大学大学大学大学大学大学大学大学大学大学大学大学大学大学大学大
		*
122	4	DEBEGLINEH: PROCEDURE PUBLIC;
123	N	IF HARDCOPY
		THEN CALL DENEMLINEH;
125	N	ELSE CALL DEBEGLINE;
126	N	RETURN
127	N	END DEBEGLINEH;
		*
		**********************

DBSYNCPR

# SEND SYNC PRINT MSG TO MODULE IN DECRT

PROCEDURE PUBLIC DBSYNCPR: 128 (MSGTEXT BASED A1) (1) BYTE; ğ N 129

/\* SEND PRINTSYNC MSG WITH TELLBACK REQUEST \*/ IF DBTELLBACK N

130

DBTELLBACK = FALSE; THEN DO:

DBMN18(3) = B1 + 4; DEMN18(0) = DECRT;

IF NOT SENDY, DBMN18(0)) THEN RETURN

/\* SEND PRINTSYNC MSG \*/

138 140

DBMM11(0) = DBCRT ELSE DO;

DBMM11(3) = B1 + 4;

IF NOT SEND(. DBMN11(0)) THEN RETURN.

DO B2 = 0 TO B1; B1 = B1 - 1;

MSGDATA(B2) = MSGTEXT(B2),

END

145 146 147

141 143 144 144

/\* SEND MSG \*/

RETURN

END DBSYNCPR. NN 148 149

<del>单件的基础的表现的表现的表现的表现的表现的表现的是的是有的的的是是是有的的的是是是有的的的是是是有的的的是是是有的的的的。</del> 

DBSYNCPRH

IF HARDCOPY = TRUE : PRINT ON LP PRINT DEBUG OUTPUT

FALSE: TYPE ON CRT

IF HARDCOPY THEN DBCRT = LP; B3 = DBCRT;

CALL DBSYNCPR;

DBCRT = 83; RETURN

END DESYNCPRH:

NNNNNN

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SEND INPUT REQUEST MSG TO MODULE IN DBCRT

DBINPREQ

PROCEDURE PUBLIC: DBINPREQ:

158

DBMN12(0) = DBCRT; /\* KMN \*/ O 159

N

160

IF NOT SEND(. DBMN12(8)) THEN RETURN;

ILER
금
0 0 0 0
ខ
98
ŧ
٣

MSGDATA(0) = B1;  /* SEND MSG, B1 CONTROLS 'ROLL SCREEN' */  END DBINPREQ;  /*  ********************************	SEND PROMPT AND REQUEST INPUT NSG TO CRT  **  *******************************	DBTBCODE = 0; DBMSWITCH = 0; DBFUNCTION = 0FFH; DBTELLBACK = FALSE; CALL DBNEWLINE; B1 = LENGTH(PROMPTMSG); CALL DBSYNCPR;  -* SEND MSG */ B1 = FALSE; CALL DBSYNCPR;	/* SEND INP REQ MSG, NO ROLL AFTER INPUT */ CRINPUT = DEBUGCOMMAND; /* DETERMINE NEXT INPUT */ RETURN; END DBPROMPT;
0 00	н	00000000 00	0 00
162 163 164	165	166 167 168 170 171 172 173	176 177 178

		**************************************
		***************************************
		SEND '?' AND PROMPT CHARACTER
		** ***********************************
179	н	*/ DBILLEGAL: PROCEDURE PUBLIC;

/\* SEND ILLEGAL MSG \*/ /\* SEND PROMPT MSG \*/ A1 = .ERRMSG(0); B1 = LENGTH(ERRMSG); CALL DBSYNCPR; CALL DEPROMPT, END DBILLEGAL; RETURNS NNN N NN 186 181 182 183 184

TERMINATE DEBUG / PROCESS T CONMAND

DETERM

PROCEDURE PUBLIC:

DBTERM:

186

œ
1LER
_
Ē.
*
COMP
_
Ø
88
1
Ė
1
ş
-

CRTINPUT = INITDEBUG;	/* NEXT DB INPUT IS INITIALIZATION */	B1 = LENGTH(TERMMSG);	R1 = . TERMMSG(0);	CALL DBSYNCPR;	/* SEND TERMINATE TEXT */	DBMN14(0) = DBCRT;	IF NOT SEND(. DBNN14(0)) THEN RETURN:	/* SEND TERMINATE 1/0 MSG TO CRT */	DEBUG = FALSE;	RETURN	END DBTERM:	
												*/
N		N	N	N		N	N		N	N	Ø	
187		188	189	196		191	192		194	195	196	

### PROCESS DEBUG REQUEST

 DEREG

\*\*

Ø

#### PL/M-86 COMPILER

B1 = LENGTH(ERRMSG1); CALL DBSYNCPR; /* SEND ERROR MSG */	CALL DBTERM; /* TERMINATE 1/0 */	RETURNS	END;	CALL DBPROMPT;	/* SEND PROMPT CHARACTER AND INPUT REQUEST */	CRIINPUT = DEBUGCONMAND;	/* NEXT INPUT IS DEBUG COMMAND */	DEBUG = TRUE;	RETURN	END DBREQ;	*	*************************************	**************************************	**	DBEXTREG	EXTERNAL DEBLIS PROJECT
мм	M	M	M	N		N		N	N	N						
284	586	267	208	289		210		211	212	213						

### EXTERMAL DEBUG REQUEST

PROCEDURE PUBLIC: DBEXTREQ: 214 DBCRT = MSGDATA(0);

/\* MODULE NUMBER OF SRT MODULE \*/
CALL DBREQ;
RETURN;
END DBREXTREQ;
END DBNOD9; N 215

CRT MODULE

4

ISIS-II PL/M-80 V3.0 COMPILATION OF MODULE CRIMODULE NO OBJECT MODULE REQUESTED COMPILER INVOKED BY: PLM80 :F1:CS.SRC NOOBJECT PAGEWIDTH(80) PAGELENGTH(35)

CRTMODULE:  **********  ***  ***  ***  ***  ***	RTMODULE: DO;  ***********************************
11 1 PCL CR	CL CRIDATAIN LIT '0EEH', /* DATA INPUT */ CRISTATIN LIT '0EFH', /* STATUS INPUT */

/* DATA QUIPUT */ /* COMMAND QUIPUT */ /* MODE CONTROL WORD */ /* COMMAND CONTROL WORD */ /* RESET CRT */	**************************************	/* HOME CURSOR */ /* FORWARD CURSOR */ /* BACK CURSOR */ /* UP CURSOR */ /* CARRIAGE RETURN */	/* LINE FEED */ /* CLEAR SCREEN */ /* START X/Y ADDRESSING */ /* ROLL MODE */ /* DEBUG (CONTROL D) */ /* INTERRUPTION (CONTROL I) */ /* TERMINATE PER. DEBUG FUNCTION (CONT	/* CLEAR LINE */ /* BEGIN OF LINE */ /* LINE FOR ASYNCHRONOUS PRINTS */ /* INPUT LINE */ /* UP ARROW */ /* BLANK */ /* BRCK SPACE - DELETE LAST CHARACTER
CRIDATADUT LIT 'ØEEH', CRICMDOUT LIT 'ØEFH', MCW LIT 'Ø1001010B', CCW LIT '00100111B', RESET LIT '01000000B';	**************************************		LF LIT '88H', CLEAR LIT '1EH', ADRXY LIT '4CH', ROLL LIT '1DH', BELL LIT '7', DEBUGMODE LIT '4', INT LIT '9', TERMPER LIT '14H',	- ROL T> */ CLRLINE LIT '17H', BOL LIT '96', ASYNCLINE LIT '102', INPLINE LIT '119', ARROW LIT '5EH', BLANK LIT '''',

M

which are some and an end are a substantial in an area of the

```
/* DELETE ENTIRE INPUT (RUB OUT) */
                                                                                                                                                                                                                                                                                                                                                                                                                                             SEND TO MODULE CONNECTED TO CRT IF INPUT IS 'INT' */
                                                                                                               /* LINE LENGTH */
/* LENGTH OF OUTPUT BUFFER */
                                                                                                                                                                                                                                                                                                                                1ST DATA BYTE: TRUE - REQUEST ACKNOWLEDGED
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         /* TERMINATE PERIODIC DEBUG FUNCTION */
                                                                                                                                                                                                                                                                                                                                                             FALSE OTHERWISE */
                                                                                                                                                                                                                                                                                                                                                                                                                                                                          CSMN23(4) BYTE INITIAL (DB, CS, 23, 4),
                                                                                                                                                                 MESSAGE CONTROL BLOCKS
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    CSMN26(4) BYTE INITIAL (0, CS, 26, 5);
/* TELLBACK MSG */
                                                                                                                                                                                                                                                                          CSMN21(4) BYTE INITIAL (0, CS, 21, 5),
                                                                                                                                                                                                                                                                                                                                                                                      CSMN22(4) BYTE INITIAL (0, CS, 22, 4),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                CSMN25(4) BYTE INITIAL (0, CS, 25, 4),
                                                                                                                                                                                                                   DCL CSMN28(4) BYTE INITIAL (0, CS, 20, 0),
                                                                                                                                                                                                                                                                                                       /* INPUT REQUEST TELL BACK MSG
                                                                                                                                                                                                                                               /* TRANSFER INPUT TEXT MSG */
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   /* START DEBUG MSG */
                                                                                                                                                                                                                                                                                                                                                                                                                    /* TERMINATE 1/0 MSG
 LINELENGTH LIT 79",
                       CRTOUTLEN LIT '200',
                                                  DELINPUT LIT '7FH';
                                                                                                                                                                      **
                                                                                                                                                                                                                        4
                                                                                                                                                                                                                      13
```

IMPLINE), /\* ROLL SCREEN.CLEAR LINE AFTER ASYNCLINE, RETURN TO

DCL ROLLSCR (8) BYTE INITIAL (CR. ADRXY, BOL, ASYNCLINE+1, CLRLINE, ADR

XY, BOL

ત

14

CONSTANT DATA

```
DELTEXT(4) BYTE INITIAL (ADRXY, 0, IMPLINE, CLRLINE),
BEGIN OF INPUT LINE */
```

/\* GO BACK TO BEGIN OF LAST INPUT,

CLEAR REST OF LINE \*/

NEWL(2) BYTE INITIAL (1, ' '), BEGL(4) BYTE INITIAL (0, ADRXY, BOL, INPLINE),

ASYNC1(10) BYTE INITIAL (ARROW, ADRXY, BOL, ASYNCLINE, CLRLINE,

BELL, \*\*\* '),

/\* START ASYNCHRONOUS PRINT \*/

ASYNC2(3) BYTE INITIAL (ADRXY, 0, INPLINE), /\* RETURN FROM ASYNCHRONOUS PRINT \*/

INITCRI(25) BYTE INITIAL (CLEAR, ROLL, ADRXY, BOL, ASYNCLINE,

BELL, '\*\*\* SYSTEM START', ADRXY, BOL, INPLINE);

/\* INITIALIZE SCREEN AT SYSTEM START \*/

\*\*

\*\*

VARIABLE DATA

CSBEGDATA BYTE, INACTMOD BYTE,

4

15

/\* MODULE WITH CURRENT INPUT ACTIVE \*/

/\* CRT INPUT BUFFER \*/ CRTINCLINELENGTH) BYTE,

CRIGOT (CRIGOTLEN) BYTE.

/\* CRT OUTPUT BUFFER \*/ CINX, OUTX) BYTE,

/\* POINTER TO CRTIN AND CRTOUT (NEXT TO FILL) \*/ CURSORINP BYTE,

/\* CURSOR POSITION ON INPUT LINE \*/

(CURSORSAVE, INXSAVE) BYTE,

/\* SAVE CURSORINP AND INX AT BEGIN OF INPUT \*/

OUTRCTIVE BYTE,

/\* TRUE IF OUTPUT ACTIVE \*/

INPREQUEST BYTE,

/\* TRUE IF INPUT REQUEST MSG RECEIVED \*/

CUTPTR BYTE,

/\* POINTER TO NEXT CHARACTER TO PRINT \*/

ROLLSCREEN BYTE.

/\* TRUE IF SCREEN IS TO ROLL AFTER INPUT \*/

/\* IF 0 THEN FIRST CALL OF CSOUTPUT, CSINPUT \*/
(CSOUTADE, CSINADE) ADDRESS, (CSOUTFL, CSINFL) BYTE.

/\* ADDRESS OF PROCEDURES CSOUTPUT/CSIMPUT \*/
(CSINX, CSOUTX) BYTE,

/\* CS INDICES FOR PRIORITY CALLS \*/

CHAR BYTE,

/\* INPUT CHARACTER \*/

DEBUG BYTE,

/\* TRUE IF DEBUG MODE \*/

CSTELLBRCK BYTE,

/\* TRUE IF TELLBACK MSG REQUIRED AFTER SYNC OUTPUT \*/ CSTBCODE BYTE,

/\* REQUIRED CODE FOR TELLBACK MSG \*/

CSENDDATA BYTE,

/\* END OF VARIABLE DATA \*/

w

\$EJECT \$NOLIST /* **********************************	CONVERT INPUT TO XY-ADDRESSING ** *********************************	DCL (A.B) BYTE;	<pre>A = B + 95, IF B &lt; 33 THEN RETURN A; A = A - 64, IF B &lt; 65 THEN RETURN A; A = A - 64, RETURN A; END CSCONVXY;</pre>	**************************************
	н	N	0000000	1 1
	163	164	185 186 188 111 112 113	

~

#### PL/M-80 COMPILER

# MOVE CHARACTERS FROM MSG INTO CRT OUTPUT BUFFER

CSMOVEOUT: PROCEDURES 114 B2 = MSG(ML) - 5;DO WHILE  $B1 \leftarrow B2$  AND  $OUTX \leftarrow CRTOUTLEN;$ NNMMMMNN 115 116 117

CRIGUICOUTX) = MSGDRIR(B1);

OUTX = OUTX + 1

B1 = B1 + 1;

END;

RETURN

END CSMOVEOUT;

## PACK INPUT INTO CRT OUTPUT BUFFER

PACKCRTOUT

PACKCRTOUT: PROCEDURE(B); 123

B BYTE; S N

124

IF OUTX > CRIOUTLEN THEN RETURN; /\* OVERFLOW \*/

N

125

127

CRTOUT(OUTX) = B;N

121 122

œ

OUTX = OUTX + 1;  RETURN; END PRICKORTOUT;  /* *********************************	PACKROLL  **  ***  ***  ***  ***  **  **  **	DO B1 = 0 TO LAST(ROLLSCR); CALL PACKCRTOUT(ROLLSCR(B1)); END; CURSORINP = 1; CURSORSAVE = 1; RETURN; END PACKROLL;	/* ***********************************
000	4	000000	
128 129 130	13	132 133 135 135 137	

CONTINUE CRT OUTPUT
PRIORITY PROCEDURE, CALL SCHEDULED BY INT 3
PROCESS LEVEL 3 CRT OUTPUT INTERRUPT

Q,

#### PL/M-80 COMPILER

CSOUTPUT: PROCEDURE;	OUTPUTTEST ADDRESS;	IF CSOUTFL = 0 THEN DO:	/* FIRST CALL, FIND BEGIN ADDRESS OF CSOUTPUT */	CSOUTROR = PROCROR;	RETURN;	END;	OUTPUTTEST = OUTPUTTEST + 1;	IF OUTPTR >= OUTX THEN	/* NO MORE CHARACTERS FOR OUTPUT */	000;	0.01PTR = 6; 0.07X = 6;	OUTACTIVE = FALSE;	IF CSTELLBACK THEN	/* TELLBACK MSG REQUIRED */	D0;	CSTELLBACK = FALSE;	CSMN26(0) = INACTMOD;	IF NOT SEND(, CSNN26(0)) THEN RETURN:	/* SEND REQUIRED MSG AND TELLBACK CODE */	END;	RETURN	END;	OUTPUT(CRIDATROUT) = CRIOUT(OUTPIR);	/* OUTPUT NEXT CHARACTER */	OUTPTR = OUTPTR + 1;	RETURN	END CSOUTPUT;	
CSOL	DCL																											*
स	N	aa		M	M	M	01	01		O	Μ	M	M		M	4	4	4		4	M	Μ	Ø		0	Ø	Ø	
139	140	141 142		143	144	145	146	147		148	149	151	152		153	154	155	156		158	159	169	161		162	163	164	

**************************************	STARTOUT	START CRI OUTPUT IF NO OUTPUT ACTIVE	** ***********************************	*/ STARTOUT: PROCEDURE;	TE CHITACTIVE THEN BETTIENS	CBL CSCHITCHT:	OUTBOTIVE = TRUE;	RETURN	END STARTOUT:	*	*************************************	老者表示是非常是是是非常是非常是非常是非常是非常是非常是非常是非常是非常是非常是非常是非	**	O I EKATO	TERMINATE 1/0	**	**************************************	/*	TERMIO: PROCEDURE;	IF INACIMOD = GFFH THEN RETURNS	/* NO MODULE WITH ACTIVE I/O */	CSMN22(0) = INACTMOD;	IF NOT SEND(, CSMN22(8)) THEN RETURN
				<del>स</del>	0	10	10	N	2										7	8		N	N
				165	166	160	169	176	171										172	173		175	176

/* SEND TERMINATE MSG */ INACTMOD = ØFFH, INPREQUEST = FALSE, DEBUG = FALSE, CALL PACKROLL; CALL STARTOUT; RETURN, END TERMIO,	/* ***********************************	CONTINUE CRT INPUT PRIORITY PROCEDURE , CALL SCHEDULED BY INT 3 PROCESS LEVEL 3 CRT INPUT INTERRUPT ** *********************************	CSINPUT: PROCEDURE;	DCL INPUTTEST ADDRESS;	IF CSINFL = 0 THEN  /* FIRST CALL, FIND BEGIN ADDRESS OF CSINPUT */	DO; Cettagos = process	RETURN	END;	INPUTTEST = IMPUTTEST + 1;	CHAR = INPUT(CRIDATAIN);
0000000			त	N	N	OI N	ı M	M	N	N
178 179 180 181 182 183			185	186	187	188	198	191	192	133

/* GET NEXT CHARACTER */	IF CHAR = INT THEN	/* CRT INTERRUPTION */	DQ;	CALL TERMIO;	GO TO INITIN:	END;	IF CHAR = DEBUGNODE THEN	/* START DEBUG REQUEST */	00;	IF DEBUG THEN GO TO ILLCHK;	/* ALREADY IN DEBUG MODE */	CALL TERMIO;	IF NOT SEND(, CSNN23(0)) THEN GOTO INITING	/* SEND DEBUG REQUEST MSG */	DEBUG = TRUE;	INACTMOD = DB;	GO TO INITIN:	END;	IF CHAR = TERMPER THEN	/* TERMINATE PERIODIC DEBUG FUNCTION */	DO:	IF NOT DEBUG THEN GO TO ILLCHK:	/* ILLEGAL INPUT */	CSMN25(0) = INACTMOD;	B1 = SEND(, CSMN25(8));	/* SEND TERM PER FUNCTION MSG */	GO TO INITIN:	END;	IF OUTROTIVE OR NOT INPREQUEST THEN RETURNS	/* OUTPUT ACTIVE OR NO INPUT REQUEST */	IF CHAR = DELCHAR THEN
	N		N	M	M	M	N		N	M		M	Μ		M	M	M	M	N		N	M		M	M		M	M	N		N
	194		195	196	197	198	199		266	261		263	204		206	282	268	269	216		211	212		214	215		216	217	218		228

#### PL/M-80 COMPILER

/* DELETE LAST CHARACTER */ DO:	THEN CALL PACKCRTOUT(BELL)	ELSE DO;	CALL PACKCRTOUT(BC);	CALL PACKCRTOUT(CLRLINE);	INX = INX - 1	CURSORINP = CURSORINP - 1;	END;	GO TO INITOUT;	END;	IF CHAR = DELIMPUT THEN	/* DELETE ENTIRE INPUT */	00;	CURSORINP = CURSORSAVE;	INX = INXSAVE	DELTEXT(1) = CSCGNVXY(CURSORINP);	DO $B1 = 0$ TO LAST(DELTEXT);	CALL PACKCRTOUT(DELTEXT(B1));	END;	GO TO INITOUT;	END;	IF CHAR = CR THEN	/* END OF INPUT */	00;	CSNN28(8) = INACTMOD;	/* SET RECEIVING MODULE */	CSMN20(3) = INX + 3;	/* MSG LENGTH */	INPREQUEST = FALSE;	IF SEND(, CSMN20(0))
N i	4	M	4	4	4	4	4	M	M	N		N	M	M	M	M	4	4	M	M	N		N	M		M		M	M
224	777	224	225	226	227	228	229	238	231	232		233	234	235	236	237	238	239	248	241	242		243	244		245		246	247

```
IF CHAR < 20H OR CHAR > 5AH OR CURSORINP >= LAST<CRTIN
                                                                                                                                                                                                                       /* PRINT '?' INSTERD OF INPUT CHARACTER */
                                                                                       /* RECEIVING MODULE NO LONGER ACTIVE */
                                       MSGDATA(B2) = CRTIN(B1);
/* SEND INPUT TEXT MSG */
                                                                                                                                                                                                                                                                                                       CURSORINP = CURSORINP + 13
                             DO B1 = 1 TO INX - 1;
                                                                                                                                                                                                                                                                         CALL PACKCRTOUT(CHAR);
                                                                                                                                                                                                                                 CALL PACKCRTOUT(7273)
                                                                             ELSE INACTMOD = OFFH;
                                                                                                                                                                                                                                           CALL PACKCRTOUT(BC);
                                                                                                                                                                                                   /* ILLEGAL INPUT */
                                                                                                                                                                                                                                                                                   CRIINCINX) = CHAR
                                                                                                          IF ROLLSCREEN THEN
                                                B2 = B2 + 1;
                                                                                                                                                                                                                                                                                             INX = INX + 1;
                                                                                                                              CALL PACKROLL;
                                                                                                                                         GO TO INITOUT;
                                                                                                                                                           GO TO INITIN:
END:
                  B2 = 6;
                                                          END;
         THEN DO;
                                                                                                  INX = 1;
                                                                                                                                                                                                                                                                ELSE DO;
                                                                                                                                                                                                              THEN DO;
                                                                                                                                                                                                                                                       EZ
Ç
                                                                                                                                                                                ILLCHK:
                                                                                                                                                                                                                                   MMMMMMMMM
                                                                                                  WWW4
                                                                                                                                         4
                                                                                                256
257
258
259
268
261
261
                                                                                                                                                                                                                                  266
267
268
269
273
273
273
274
274
                  252
252
253
253
253
253
253
253
253
253
                                                                                                                                                            262
263
264
```

# PL/M-80 COMPILER

INITIN:	* * *	PRINT ASYNCHRONOUS TEXT (MN 10)  ( MAX 1 LINE )  **  *******************************	CSPRINTASYNC: PROCEDURE; DO B1 = 0 TO LAST(ASYNC1);	CALL PACKCRTOUT(ASYNC1(B1));	B1 = 0; Cel 1 CCMONECUT:	CALL CARACTORY /* PACK MSG INTO OUTPUT BUFFER */ ASYNC2(1) = CSCONVXY(CURSORINP);	/* RETURN CURSOR TO LAST INPUT POSITION */	CALL PACKCRTOUT(ASYNC2(B1));	END;	RETURN;	END CSPRINTASYNC;	*
0000			ત લ	MM	000	1 0		ım	MC	v N	Ŋ	
275 276 277 278			279	287	283	282	986	287	888	298	291	

CSPRINTSYNC

(MN 11 AND MN 18) PRINT SYNCHRONOUS TEXT PRINT ON INPUT LINE ROLL SCREEN AFTER OUTPUT IF MSGDATA(B) = TRUE

CSPRINTSYNC: PROCEDURE

292

/\* MSG NOT FROM CORRECT MODULE \*/ IF MSG(SMN) <> INACTMOD THEN N

CALL ILLEGALMSG;

RETURN

NMMMNN

2882 2882 2882 2882 2882 2882

END

CALL CSMOVEDUTA 1; **B**1

/\* MOVE CHARACTERS FROM MSG TO CRT OUTPUT BUFFER \*/

IF MSGDRTR(8)

O

366

THEN CALL PACKROLL,

ELSE CURSORINP = CURSORINP + MSG(ML) /\* ROLL SCREEN AFTER OUTPUT \*/

CALL STARTOUT;

RETURN

NNNN

362 363

END CSPRINTSYNC;

\*

CSINPREQ

```
INPUT REQUEST FROM MODULE CONNECTED TO CRT (MN 12)
                                                                                            IF MSGDATA(0) = TRUE : ROLL SCREEN AFTER INPUT
                                               SEND INPUT UP TO CR TO MODULE INACTMOD
```

PROCEDURE; CSINPREQ: ત 306

N

307

/\* MSG NOT FROM CORRECT MODULE \*/ IF MSG(SMN) <> INACTMOD THEN

CALL ILLEGALMSG

RETURNS

CURSORSAVE = CURSORINP;

NNNNNNNN

313 314

INXSRVE = INX;

ROLLSCREEN = MSGDATA(0); INPREQUEST = TRUE;

/\* TRUE IF NEW LINE AFTER INPUT \*/

RETURN

NN

316

END CSINPREQ;

ACTIVATE INPUT FOR SENDING MODULE

(MN 13)

CSINPACTIVATE

\*\*

**************************************	CSINPACTIVATE: PROCEDURE;	CSMN21(0) = MSG(SMN); IF NOT SEND( CSMN21(0)) THEN RETURN; /* SEND ACKNOWLEDGE MSG BACK */	IF INACTMOD <> OFFH THEN MSGDATA(0) = FALSE;  /* CPT NOT FREE */	ELSE DO;	MSGDATA(0) = TRUE; INACTMOD = MSGCSMN):	INPREQUEST = FALSE;	END	RELUKN; END CSINPACTIVATE;	*	**************************************	CSNEMLINE	POSITION CURSOR AT BEGIN OF NEW LINE ( MN 16 )	*************************************	CSNEWLINE: PROCEDURE;	MSG(ML) = 6; RDRMSGDATA = .NEWL(0); CALL CSPRINTSYNC;
	4	NN	N	N	M M	ı M	M	NN						4	200
	318	319	322	45	325	322	328	336						331	3433

œ
ER
_
ĭĽ
<u>a</u>
SOME
O
O
-
88
É
÷
ì

RETURN; END CSNEWLINE; /*  *********************************	POSITION CURSOR AT BEGIN OF INPUT LINE < MN 17 > ** *********************************	MSG(ML) = 8; ADRMSGDATA = . BEGL(0); CALL CSPRINTSYNC; CURSORINP = 1; CURSORSAVE = 1; RETURN; END CSBEGLINE;	/* ***********************************	INITIALIZE CRT MODULE (MN 00)  **  ***  ***  ***  **  CSSTART: PROCEDURE;
NN	н	000000		н
332	337	338 340 340 341 341 341		345

IF RESTART THEN OUTPUT(CRICNDOUT) = RESET; /* RESET USART IF NO SYSTEM RESET */	CALL CLEARDATAC CSBEGDATA, CSENDDATA); /* CLEAR VARIABLE DATA */	CALL CSINPUT;	CALL CSOUTPUT;	INFL = 1;	/* FIND PROCEDURE START ADRRESSES */	CALL UPDSTAT(CS, 00001011B);	/* SET MODULE STATUS */	DO B1 = 0 TO LAST(INITCRT);	/* INITIALIZE SCREEN */	CRTOUT(B1) = INITCRT(B1);	EMD;	OUTX = LENGTH(INITCRT);	INX = 1; CURSORINP = 1;	CURSORSAVE = 1;	ROLLSCREEN = TRUE;	INACTMOD = GFFH;	INPREQUEST = FALSE;	DEBUG = FALSE;	CSOUTX = ENTERPRIOR(CSOUTADR);	CSINX = ENTERPRIOR(CSINADR);	/* ENTER PRIORITY CALLS */	OUTPUT(CRICMDOUT) = MCW;	/* MODE CONTROL WORD */	B1 = 0;	OUTPUT(CRICMDOUT) = CCW;	/* COMMAND CONTROL WORD */	B1 = ENTERINT(4, CSINX);	B1 = ENTERINT(5, CSOUTX);	/* ENTER INTERRUPTS ON LEVELS 3 AND 4 */
N	N		2			2		2											N			2			2		N		

# PL/M-80 COMPILER

## PL/M-80 COMPILER

END;	INACTMOD = MSG(SMN); /* MN 15 */		CALL CSBEGLINE; /* MN 17 */	DO; /* MN 18 */	CSTELLBACK = TRUE; CSTBCODE = MSGDATA(0);	ADRMSGDATA = ADRMSGDATA + 1,	MSG(ML) = MSG(ML) - 1	CALL CSPRINTSYNC;	END;	END;	RETURN	END MSGENTØ6;	
													END;
4	M	M	M	M	4	4	4	4	4	M	N	N	
391	392	393	394	395	396	398	399	466	461	462	463	404	465

# MODULE INFORMATION:

CODE AREA SIZE	11	= 05ABH	14510
VARIABLE AREA SIZE = 0186H	11	<b>0186H</b>	3990
MAXIMUM STACK SIZE = 866CH	11	ВВВСН	120
1006 LINES READ			
8 PROGRAM ERROR(S)			

# END OF PL/M-80 COMPILATION

LINE PRINTER MODULE

ISIS-II PL/M-80 V3.0 COMPILATION OF MODULE LPMOD NO OBJECT MODULE REQUESTED

PLM80 : F1:LP. SRC NOOBJECT PAGEWIDTH(80) PAGELENGTH(35) COMPILER INVOKED BY:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* LINE PRINTER MODULE SUBSTITUTIONS MODULE IDENTIFICATION: LP MODULE NUMBER LP DATA ô \* NOLIST LPMOD: \*\* \*\* \*\* H

/\* LP DATA OUTPUT PORT \*/
/\* LP COMMAND OUTPUT PORT \*/ LPDATROUT LIT 'OFAH', LPCMDOUT LIT '0FBH',

4

N

```
NONOPMSG (26) BYTE INITIAL ('LINE PRINTER NOT CONNECTED'), ENDCONST BYTE;
/* LP STATUS INPUT PORT */
/* OUTPUT BUFFER LENGTH */
                                                                                                                                   /* 0 - FIRST CALL OF PROCEDURE LPOUTPUT */
                                                                                                                                                                                                        LPMN18 (4) BYTE INITIAL (CS. LP. 18, 8),
                                                                                                                                                                                                                                              LPMN26 (4) BYTE INITIAL (0, LP, 26, 5),
                                                                                                                                                                                                                                                                                    LPMN28 (4) BYTE INITIAL (0, LP, 28, 4),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      /* START ADDRESS OF LPOUTPUT */
                                                                                                                                                                                                                                                                                                                                                                                                                                                                           /* BEGIN OF CLEAR REGION */
                                                                                                                                                                                                                                                                                                       /* LP NOT CONNECTED MSG */
                                                                                                                                                                                                                                                                                                                                                                                                                      VARIABLE DATA
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        LPOUTBUF(LPOUTBUFLEN) BYTE, /* LP OUTPUT BUFFER */
                                                                                                                                                                      CONSTANT DATA
                                                                                                                                                                                                                             /* ASYNC PRINT MSG */
                                                                                                                                                                                                                                                                   /* TELLBACK MSG */
 LPSTATIN LIT '0FBH',
LPOUTBUFLEN LIT '150',
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   LPOUTPUTADE ADDRESS,
                                     FORMFEED LIT '8CH',
                                                                                          ENDSUBST LIT '8';
                                                                                                                                                                                                                                                                                                                                                                                                                                                          LPCLEARBEG BYTE,
                                                       CR LIT '8DH',
LF LIT '8RH',
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              LPOUTFL BYTE,
                                                                                                                                                                                                                                                                                                                                                                                                                                         $¢
                                                                                                                                                                                       $6
                                                                                                                                                                                                                                                                                                                                                                                                                      *
                                                                                                                                                     **
                                                                                                                                                                       **
                                                                                                                                                                                                                                                                                                                                                                                                      **
                                                                                                                                                                                                           99
                                                                                                                                                                                                                                                                                                                                                                                                                                                            166
```

<LPOUTX, LPOUTPTR> BYTE,
/\* INDICES FOR LP OUTPUT BUFFER
LPOUTX - NEXT TO FILL

LPTELLBACK BYTE,

/\* TRUE IF TELLBACK MSG REQUESTED AFTER OUTPUT \*/

LPTBCODE BYTE,

/\* REQUESTED CODE FOR TELLBACK MSG \*/
LPPRX BYTE,

/\* LP PRIORITY INDEX \*/ LPOUTACTIVE BYTE, /\* TRUE IF OUTPUT ACTIVE \*/

LPMODSAVE BYTE,

/\* MODULE NUMBER OF PRINTING MODULE \*/ LPCLEAREND BYTE;

322 a

# PL/M-80 COMPILER

<ul> <li>EJECT</li> <li>/*</li> <li>************************************</li></ul>	LPOUTPUT	CONTINUE LINE PRINTER OUTPUT PRIORITY PROCEDURE , CALL SCHEDULED BY INT 3 TO PROCESS LEVEL 3 LP OUTPUT INTERRUPT	** ***********************************	LPOUTPUT: PROCEDURE;	IF LPOUTFL = 0 THEN /* FIRST CALL, FIND ADDRESS OF LPOUTPUT */	00;	LPOUTPUTADR = PROCADR;	RETURN	EMD;	IF LPOUTPTR >= LPOUTX THEN  /* NO MODE CHOROCTEDS TO DOINT */	DO:	LPOUTPTR = 0, LPOUTX = 0;	LPOUTACTIVE = FALSE;	IF LPTELLBACK THEN  /* TELLBACK MSG REQUESTED */	<b>D0</b>	LPTELLBACK = FALSE;	LPMN26(RMN) = LPMODSAVE;
				4	N	N	MM	m	M	W.	~	M	M	M	M	4	4
				161	162	163	164	106	187	108	169	116	112	113	114	115	116

ທ

IF NOT SEND(.LPMN26(0)) THEN RETURN; MSGDATA(0) = LPTBCODE; /* SEND TELLBACK MSG WITH REQUESTED CODE */	ãã	END; OUTPUT <lpdataout> = NOT LPOUTBUF<lpoutptr>; /* OUTPUT NEXT CHARACTER */</lpoutptr></lpdataout>	LPOUTPTR = LPOUTPTR + 1; RETURN; END LPOUTPUT;	/* ***********************************	START LINE PRINTER OUTPUT  ** ******************************	*/ LPSTARTOUT: PROCEDURE;	IF LPOUTACTIVE THEN RETURN; CALL LPOUTPUT; LPOUTACTIVE = TRIE;	RETURN; END LPSTARTOUT;	/* ***********************************
4 4	4 W	m N	000			ત	000	100	
117	120	122	124 125 126			127	128	132	

#### LPPACK

BUFFER
Ш
L
F
灵
ш
_
5
2
DUTPUT
$\supset$
3
0
INTO
-
~
1
=
11
ã
Ø.
Œ
CHARACTER
Ü
4
.,
ž
×
à
-

LPPACK: PROCEDURE (CHAR) BYTE,

134

135

CHRR BYTE; Bo IF LPOUTX > LAST(LPOUTBUF) THEN DO;

/\* BUFFER OVERFLOW \*/
CALL SYSMON(STACKPTR, 5, 0);

RETURN FALSE:

MMMN

140

138

ELSE DO;

LPOUTBUF (LPOUTX) = CHAR; /\* PRCK CHRRRCTER \*/

LPOUTX = LPOUTX + 1;

MMMNN

146

RETURN TRUE; END LPPRCK;

POSITION PRINT HEAD AT BEGIN OF NEW LINE

LPNEWLINE

325

N

~

	PROCEDURE;
	PNEWLINE
*	LPN

147

RETURN	RETURN				
THEN	THEN				
IF NOT LPPACK(CR)	IF NOT LPPACK(LF)	CALL LPSTARTOUT;	RETURN	END LPNEWLINE;	
					*/
8	N	N	2	N	
148	156	152	153	154	

POSITION PRINT HEAD AT TOP OF NEW PAGE

LPNEWPAGE

	***	
	***	
	***	
	****	
	***	
	****	
	***	
	****	
	***	
	****	
	***	
	****	
	****	
	****	
	****	
	***	
*	***	*

1 LPNEWPAGE: PROCEDURE;

155

2			
EN RETUR			
IF NOT LPPACK(FORMFEED) TH	CALL LPSTARTOUT;	RETURN	END LPNEWPAGE;
2	8	2	2
156 2	158	159	169

 PROCESS PRINT MSG (MN 11) IF MSGDATA(0) = TRUE : PRINT HEAD AT BEGIN OF NEW LINE AFTER 0

LPPRINT

œ

×	
ILER	
SOM	
$\ddot{\circ}$	
88	
ŧ	
5	
4	

- UTPUT ** *******************************	B3 = MSG(ML) - 5;  IF B3 = 0 THEN RETURN;  D0 B2 = 1 TO B3;  /* MOVE CHARACTERS INTO OUTPUT BUFFER */  IF NOT LPPACK(MSGDATA(B2)) THEN G0 TO PR1;  /* OUTPUT BUFFER FULL */	END; IF MSGDATA(0) THEN CALL LPNEWLINE;  CALL LPSTARTOUT;  RETURA;  END LPPRINT;	**************************************	IF MSG(ML) < 7 THEN RETURN; LPTBCODE = MSGDATA(0);
स	000 M	M M M M M M M	4	NN
161	162 163 165 165	168 169 171 172 172	175	176 178

ø

œ
ILER
_
-
₾
÷
Š
J
Ø
8
-
Ę
1
Ļ
ş

/* SRVE TELLBACK CODE */	LPTELLBACK = TRUE;	ADRMSGDATA = ADRMSGDATA + 1;	MSG <ml> = MSG<ml> - 1; /* ADJUST MSG FOR PROCESS BY LPPRINT */</ml></ml>	LPMODSAVE = MSG(SMN);	/* SRVE MODULE NUMBER FOR TELLBRCK MSG */	CALL LPPRINT;	RETURN;	END LPPRINTTB;	*	**************************************	**	LPINIT	INITIALIZE LINE PRINTER	**	**************************************	*/ LPINIT: PROCEDURE;	IF (INPUT(LPSTATIN) AND 2) <> 0	THEN DO;	/* LP NOT CONECTED */	LPMNZ8(RNN) = MSG(SMN);	B1 = SEND(. LPNN28(0));	/* SEND 'LP NOT CONNECTED' MSG */	LPMN10 <ml> = LENGTH<nonopmsg> + 4;</nonopmsg></ml>	IF NOT SEND(. LPMN10(0)) THEN RETURN;	/* SEND ASYNC PRINT MSG TO CS */	DO B3 = 0 TO LAST(NONOPMSG);
	N	N	N	N		N	N	N								त	N			M	M		M	M		M
	179	180	181	182		183	184	185								186	187			189	190		191	192		194

MSGDRTR(B3) = NONOPMSG(B3);  END;  END;  ELSE CALL LPNEWPRGE;  /* LINE PRINTER CONNECTED */  RETURN;  END LPINIT;  /*  ********************************	** INITIALIZE LP MODULE ** *********************************	LPSTART: PROCEDURE; CALL CLEARDATA< LPCLEARBEG, LPCLEAREND>; /* CLEAR VARIABLE DATA */	CALL LPOUTPUT;  /* FIND PROCEDURE START ADDRESS */ CALL UPDSTAT(LP, 3);  /* SET MODULE STATUS */	LPPRX = ENTERPRIOR(LPOUTPUTADR);  /* ENTER PRIORITY CALL TO PROCESS INTERRUPT */	RETURN; END LPSTART;	/* ***********************************
44WU UU		4 0	N N	N	NN	
195 196 197 198 200		201	263	285	286 287	

**MSGENTØ5** 

11

### PL/M-80 COMPILER

\*\*

MESSAGE ENTRY OF LP MODULE

PROCEDURE PUBLIC: MSGENT05: 288 IF MSG(MN) = 0 THEN /\* START MSG \*/

209

CALL LPSTART; RETURN

IF MSG(MN) < 11 OR MSG(MN) > 18 ELSE DO CASE MSG(MN) - 11; THEN CALL ILLEGALMSG,

CALL LPPRINT; CALL LPNEWPAGE;

ILLEGALMSG; ILLEGALMSG; CALL LPINIT; CALL CALL

LPNEWLINE; LPNEWLINE; LPPRINTTB; CALL CALL CALL

216 217 218 218 228 222 222 223 223 225 225 226 227 228

END

END MSGENT05; RETURN

EMD;

330

212 213 214

LOADER

The state of the s

ISIS-II PL/M-80 V3.0 COMPILATION OF NODULE LOADER NO OBJECT NODULE REQUESTED

COMPILER INVOKED BY: PLM80 :F1:LOADER.SRC NOOBJECT PAGEWIDTH(80) PAGELENGTH(35)

LOADER: DO;

2 1 DECLARE LIT LITERALLY 'LITERALLY'

1 DECLARE DCL LIT 'DECLARE';

1 DCL CR LIT '0DH',

LF LIT '0AH', NUMSBC LIT '3', INTVECTOR LIT '3000H',

BIAS LIT '6000H', BEGINRAM LIT '3000H', 1 DCL SBC BYTE, /\* NUMBER OF SBC BEING LOADED \*/

n

(MEMINTS BASED A1) (1) BYTE, (MEMINTD BASED A2) (1) BYTE, (A1, A2) ADDRESS, (B1, B2) BYTE,

FILENAME (6) BYTE INITIAL ('LOADØ/'),
/\* FILENAMES OF FILES CONTAINING CODE FOR SBC'S \*/

/\* BYTE 0: LOADED INT LEVEL, BYTE 1:ACTUAL INT LEVEL ETC. INTSHFTTBL (10) BYTE INITIAL (3, 3, 4, 4, 5, 5, 6, 0, 7, 2),

KEY ADDRESS AT (0F1F4H) INITIAL (0ABCDH),

CORDSBC BYTE AT (0F1F0H),

START1 BYTE AT (. LOADSBC + 1), START2 BYTE AT (. LOADSBC + 2),

STARTS BYTE AT (. LOADSBC + 3),

TEXT1 (\*) BYTE INITIAL (CR, LF, LF, 'SBC LOADER', CR, LF, LF),
TEXT2 (\*) BYTE INITIAL ('SBC LOADED', CR, LF, LF),
TEXT3 (\*) BYTE INITIAL ('LOAD FILE FOR SBC 0 NOT FOUND', CR, LF, 5

# SBCRAM (4096) BYTE AT (BEGINRAM + BIAS);

PROCEDURE (P1, P2, P3, P4, P5) EXTERNAL;	(P1, P2, P3, P4, P5) ADDRESS;	END;	PROCEDURE (P1, P2, P3, P4) EXTERNAL;	(P1, P2, P3, P4) ADDRESS;	END;	PROCEDURE (P1, P2, P3, P4, P5) EXTERNAL;	(P1, P2, P3, P4, P5) ADDRESS;	END;	PROCEDURE (P1) EXTERNAL;	P1 ADDRESS;	END;	DO B1 = 0 TO LAST(SBCRAM);	SBCRRM(B1) = $\theta_i$	END;	START1, START2, START3, LOADSBC = 0;	CALL WRITE(0, TEXT1(0), LENGTH(TEXT1), A1);	VU SBC = 1 10 NOMSBC;	FILENAME(4) = SBC + 30H;	CALL LORD(, FILENAME(0), BIRS, 0, . 81, . 82);
READ:	ದ್ದ		WRITE:	DCL		LOAD:	PCL		ERROR:	ದ್ದ									
6 1	2	ري 00	9 1	10 2	11 2	12 1		14 2	15 1	16 2	17 2	18 1	19 2	20 2	21 1	22 1	23 1	24 2	25 2

/* LOAD CODE FOR SBC N */ IF R2 = 0 THEN D0;	LOADSBC = SBC; DO WHILE LOADSBC <> 0; /* WAIT UNTIL SBC TRANSFERRED CODE INTO ON-BOARD RAN */	END;	IF SBC = 1 THEN /* SFI BCTIBL INT VECTOR */	DO B1 = 0 TO LAST(INTSHFTTBL) BY 2:	R1 = SHL(INTSHFTTBL(B1),3);	R2 = SHL(INTSHFTTBL(B1+1), 3) + INTVECTOR;	DO B2 = 0 TO 7;	NEMINTD(B2) = MEMINTS(B2);	EMD;	END;	TEXT2(4) = SBC + 30H;	CALL WRITE(0, TEXT2(0), LENGTH(TEXT2), A2); /* TYPE COMPLETION MSG */	END:	ELSE DO:	TEXT3(18) = SBC + 30H;	CALL WRITE(0, TEXT3(0), LENGTH(TEXT3), . A2);	/* TYPE WARNING MSG */	EMD;	END;	START1 = 1;	/* START SBC 1 */	
Ø	мм	4	М	М	4	4	4	ຜາ	ຜາ	4	M	М	~	N	M	M		M	N	7	-	•
56	88	30	Ħ	32	M	34	32	36	37	80 M	8	4	4	4	<b>4</b>	4		45	46	47	4	)

EXTERNAL DECLARATIONS OF SYSTEM DATA

N

```
MODULES IN 1 CPTR */
                                                                                                                                                                                                                                                                                                                                                                       MODULES IN SYSTEM */
                                                                                                                                                                                                                                                                                                                                                                                                                       MAX NUMBER OF PERIODIC CALLS */
                                                                                                                                                                                                                                                                                                                                                                                                     MAX NUMBER OF PRIORITY CALLS */
                                                                                                 MN OF FIRST MODULE IN CPTR */
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     /* MN OF FIRST MODULE IN CPTR */
/* MN OF LAST MODULE IN CPTR */
                                                                                                                                                                                                                                                                                                                                        COMPUTERS */
                                                                                                                                                                                                                                                                                                                                                                                                                                                     /* INTERRUPT VECTOR */
/* NUMBER OF HOST COMPUTER */
                                                                                                                                                                                                                                                                                                                                                                                                                                       /* BEGIN OF COMMON MEMORY */
                                                                                                                                                                                                                                                                                                                                                                                        LENGTH OF MSG BUFFER */
                                                                                                                                                                                                                                                                                                                                        MAX NUMBER OF
                                                                                                                                                                                                                                                                                                                                                      MAX NUMBER OF
                                                                                                                                                                                                                                                                                                                                                                      MAX NUMBER OF
                                                                                                                                              SYSTEM DATA DECLARATIONS
                                                                                                                                                                                           DECLARE LIT LITERALLY 'LITERALLY', DECLARE',
                                                                                                                                                                                                                                                                                                                                          * *
                                                                                                                                                                                                                                                                                                                                                                       *
                                                                                                                                                                                                                                                                                                                                                                                                                       *
                                                                                                                                                                                                                                                                                                                                                                                        *
                                                                                                                                                                                                                                                                                                                                                                                                      *
                                                                                                                                                                                                                                                                                                                                                                                                                                                     INTVECTOR LIT '3000H',
                # INCLUDEC: F1: SYDATE. SRC)
                                                                                                                                                                                                                                                                                                                                                                                                                                      BEGINCM LIT '0F00H',
                                                                                                                                                                                                                                                                                                                                                                      MAXSYSMOD LIT '24',
MSGBUFLEN LIT '500',
                                                                                                                                                                                                                                                                                                                                                                                                        MAXPRIOR LIT '8',
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    FIRSTMN LIT '8',
                                                                                                                                                                                                                                                                                                                                        MAXCPTR LIT 131,
                                                                                                                                                                                                                                           TRUE LIT 'OFFH',
                                                                                                                                                                                                                                                                                                                                                                                                                       MAXPER LIT '8',
                                                UPDATE:
                                                                                                                                                                                                                                                                                                                                                         MAXMOD LIT '8',
                                                                                                                                                                                                                                                           FALSE LIT '8',
                                                                                                                                                                                                                                                                                                                                                                                                                                                                     CPTR LIT '1',
                                                                                                                                                                                                                                                                          RMN LIT '8',
                                                                                                                                                                                                                                                                                          SMN LIT '1',
                                                                                                                                                                                                                                                                                                         MN LIT '2',
ML LIT '3',
*EJECT
                                                                                                                                                                                                                                           PCP
PCP
                                                                                                                                                                 ***
                                                                                   11
                                                                                                                11
                                                                                                                                                                                                                                            M
                                                                                                                                                                                               O
```

LASTMN LIT 'FIRSTMN+7',

M

EX LIT 'FIRSTMN', /* MN OF EXEC MODULE */ LP LIT 'FIRSTMN+5', /* MN OF LINE PRINTER MODULE */ CS LIT 'FIRSTMN+6', /* MN OF CRT MODULE */ DB LIT 'FIRSTMN+7'; /* MN OF DEBUG MODULE */	*		*	DCL (81, 82, 83) BYTE EXTERNAL,			**	SYSTEM WORK VARIABLES	BUCUPPER) AND BLCLOWER) OVERLAY ADDRESS VARIABLE A4	*	DCL (ADRMSG, ADRMSGDATA) ADDRESS EXTERNAL,				MORK AREAS FOR MSG CONTROL BLOCK (MSG) AND MSG		ADRNSG AND ADRMSGDATA ARE SET BY EXEC BEFORE THE		*	DCL RESTART BYTE EXTERNAL;	*/	FALSE IF START WITH SYSTEM RESET		/*	DCL RTC (4		DCL INTMASK BYTE EXTERNAL;	
	11 11	11	11	11	11	11	11	11	11	11	II	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	u	
				त							-	•								4					4		त	
				4							17	,								9					~		ထ	

```
SUSPEND MSG, SENT BY EXEC TO MODULE TO BE SUSPENDED */
                                                                                                                                                                                                                                                 INTERNAL START MSG, SENT TO ALL MODULES IN CPTR
                                                                                                                                                                                                                                                                                                                   RESTART MSG, SENT TO MODULE TO BE ACTIVATED
/*
MASK OF CURRENTLY ACTIVATED INTERRUPTS
*/
                                                                                                                                                                        SYSTEM MESSAGES
                                          DCL DEBUGMCB (4) BYTE EXTERNAL
                                                                    MCB FOR DEBUG PURPOSES
                                                                                                                                                                                                                                                                                   SYSMN1(4) BYTE EXTERNAL,
                                                                                                                                                                                                         */
DCL SYSMNØ<4> BYTE EXTERNAL,
                                                                                                                                                                                                                                                                                                                                                   SYSMN5(4) BYTE EXTERNAL
                                                                                                                                                                                           ***
                                                                                                                                                          *
                                                                                                                                                                                                                         10
                                               Q
```

EXTERNAL DECLARATIONS OF EXECUTIVE DATA

n

the second secon

```
MODULE AUTHORIZED TO SUSPEND/ACTIVATE
                                                            MODSTAT IS AN OVERLAY FOR MODSTATUS AND COVERS
                                                                                                                                                                                                                      MODSTATUS CONTAINS THE STATUS OF ALL MODULES IN THE SYSTEM. INDEX IS MODULE NUMBER.
                                                                                                                                                                                                                                                                                                                                                                                                                              MODULE NOT AUTHORIZED TO SUSPENDA
                                                                                                                                                                                                                                                              MODSTATUS IS UPDATED BY EXEC MITH SYS MSG
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 THE BASE VARIABLE ADRSTAT IS SET AT START.
                                                                                                                                                                                                                                                                                                       0 - BIT 7 = 0: MODULE DOES NOT EXIST
                                                                                                                                                                                                                                                                                                                                                                                    MODULE MAY NOT BE SUSPENDED
                                                                                                                                                                                                                                                                                                                                                                                                          MODULE MAY BE SUSPENDED
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  (MODSTAT BASED ADRSTAT) (MAXMOD) BYTE:
                                                                                                                                                                                                                                                                                                                                                                                                                                                 ACTIVATE OTHER MODULES
                                                                                                                                                                               DCL NODSTATUS (MAXSYSNOD) BYTE EXTERNAL;
                                                                                                                                                                                                                                                                                                                                              - MODULE NOT ACTIVATED
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              THE ENTRIES FOR ONE COMPUTER.
                                                                                                                                                                                                                                                                                                                                                                  - MODULE ACTIVATED
                                                                                                                    EXECUTIVE DATA
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          OTHER MODULES
                                                                                                                                                                                                                                                                                                                         1 - MODULE EXISTS
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              ADRSTAT ADDRESS EXTERNAL,
                  # INCLUDE(:F1:EXDRTE. SRC)
                                                                                                                                                                                                                                                                                   \langle BIT 0 = LSB \rangle
                                                                                                                                                                                                                                                                                                                                                                                                                                                                          1
                                                                                                                                                                                                                                                                                                                                                 Ø
                                                                                                                                                                                                                                                                                                                                                                                       Ø
                                                                                                                                                                                                                                                                                                                                                                                                                              0
                                                                                                                                                                                                                                                                                                                                                                                                                                                                         ત
                                                                                                                                                                                                                                                                                                                                                                                                                              M
                                                                                                                                                                                                                                                                                                                                                                                       ä
                                                                                                                                                                                                                                                                                                                                               BIT 1:
                                                                                                                                                                                                                                                                                                                                                                                       BIT
                                                                                                                                                                                                                                                                                                                                                                                                                              BIT
                                                                                                                                                                                                                                                                                                        BIT
                                                                                                                                                                                                                                                                                                                            BIT
$EJECT
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                S
S
                                                                                                                                           ***
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 11
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 12
                                                                                                                                                                                 77
```

# PL/M-80 COMPILER

*	DCL BEGABSDATA BYTE EXTERNAL,	EXTMSGLOCK BYTE EXTERNAL,	EXTMSG BYTE EXTERNAL,	EXTMSGIN BYTE EXTERNAL,	EXTMSGOUT BYTE EXTERNAL,	NUMEXTMSG BYTE EXTERNAL,	EXTLASTMSG BYTE EXTERNAL,	EXTMSGBUFFER(250) BYTE EXTERNAL,	ENDABSDATA BYTE EXTERNAL;	DCL (LOADSBC, START1, START2, START3) BYTE EXTERNAL;	DCL ILLMSG (36) BYTE EXTERNAL;	*	TEXT FOR ILLEGAL MSG TO CRT	*	DCL MONMSG (4) BYTE EXTERNAL,	MONTEXT (26) BYTE EXTERNAL;	*	SYSTEM MONITOR MSG TO CS	<b>*</b>	DCL EXTRMSG (4) BYTE EXTERNAL)	*/	EXTRACTION MSG TO DEBUG MODULE	/*	DCL SAVESTACKPTR ADDRESS EXTERNAL;	*/	STACK POINTER AT SYSTEM START	<b>*</b>	DCL EXCLEARBEG BYTE EXTERNAL;	DCL MSGEXTRACTION BYTE EXTERNAL;	*	TRUE IF MSG EXTRACTION IS ACTIVATED	
11	11	11	u	H	H	u	11	n	11	n	u	u	11	11	11	11	ti	11	11	u	11	11	11	u	u	11	H	u	11	u	11	
	4									7	4				7					ત				त्त				7	त			
	13									14	15				16					17				18				19	20			

* DCL CDCRDR RDDRESS EXTERNAL,	CDCACTIVE BYTE EXTERNAL,	INTTBL(8) BYTE EXTERNAL;	* DCL RSTADR ADDRESS EXTERNAL,	SNAPINTADE ADDRESS EXTERNAL,	RSTFL BYTE EXTERNAL;	* DCL CODESAVE ADDRESS EXTERNAL,	EXPERFL BYTE EXTERNAL,	EXPERSOR SODRESS EXTERNAL,	* EXPERX BYTE EXTERNAL;	* DCL MSGBUFFER (MSGBUFLEN) BYTE EXTERNAL,	* (MSGIN, MSGOUT) ADDRESS EXTERNAL,	* NUMMSG BYTE EXTERNAL,	* LASTMSG ADDRESS EXTERNAL;	*	* MSGBUFFER IS A CIRCULAR FIFO LIST CONTAINING THE	* MSG WHICH ARE WAITING TO BE PROCESSED.	* MSGIN POINTS TO LOCATION OF NEXT MSG TO FILL IN.	* MSGOUT POINTS TO MSG TO BE PROCESSED NEXT.	NUMMSG IS INCREMENTED WHEN A MSG IS SENT AND	<ul> <li>DECREMENTED WHEN A MSG 1S PROCESSED.</li> </ul>	* NUMMSG = 0: MSGBUFFER IS EMPTY.	- LASTMSG CONTAINS INDEX OF LAST BYTE OF LAST MSG	IN MSGBUFFER + 1.	*	<ul> <li>DCL PRIORLIST (MAXPRIOR) ADDRESS EXTERNAL,</li> </ul>	PRIORSCHEDULE BYTE EXTERNAL;	*/	- PRIORLIST CONTAINS THE ADDRESSES OF THE ACTIVATED	= PRIORITY TASKS.	<ul> <li>A PRIORITY TASK IS CALLED WHEN IT IS SCHEDULED BY</li> </ul>
"	11	11	"	"	"	"	II	11	11	"	11	11	11	11	11	11	11	11	11	11	u	H	11	u	"	u	H	u	11	a
+			4			7				ਜ															7					
22			22			23				24															25					

The same of the sa

w

36

27

PL/M-80 COMPILER

= /\*
NUMBER OF ACTIVATED PERIODICS
= \*/
DCL EXCLEMREND BYTE EXTERNAL;

53

#### INTIIAL DISTRIBUTION LIST

		No.	Copi
1.	Defense Documentation Center		2
	Cameron Station		
	Alexandria, Virginia 22354		
2.	Library, Code 0142		2
	Naval Postgraduate School		
	Monterey, California 93940		
3.	Department Chairman, Code 52		3
	Department of Computer Science		
	Naval Postgraduate School		
	Monterey, California 93940		
4.	Professor U.R. Kodres, Code 52Kr		3
	Department of Computer Science		
	Naval Postgraduate School		
	Monterey, California 93940		
5.	Professor N. Schneidewind, Code 52Ss		1
	Department of Computer Science		
	Naval Postgraduate School		
	Monterey, California 93940		

6. LCDR F. Burckhead, Code 52Bg

Department of Computer Science

Naval Postgraduate School

Monterey, California 93940

- 7. Marineamt A1 1
  2940 Wilhelmshaven
  Federal Republic of Germany
- 8. Dokumentationszentrale der Bundeswehr (See)

  Friedrich-Ebert-Allee 34

  5300 Bonn

  Federal Republic of Germany
- 9. Kapitaenleutnant Wolfgang Niemann
  Kommando Marinefuehrungssysteme
  Wibbelhofstrasse 3
  2940 Wilhelmshaven
  Federal Republic of Germany